Technická univerzita v Liberci
Fakulta strojní

DIPLOMOVÁ PRÁCE

Liberec 2003 Petr Oliva

Technická univerzita v Liberci

Fakulta strojní

Studijní obor : 23 – 40 – 8 Automatizované systémy řízení ve strojírenství
Zaměření : Automatizace inženýrských prací

Katedra aplikované kybernetiky

Laboratorní model řízení fotovoltaických článků

Control of photovoltaic panels - laboratory model

Petr Oliva

Vedoucí diplomové práce : Ing. Slavomír Němeček
Konzultant diplomové práce : Ing. Václav Lenk

ANOTACE

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní
Katedra aplikované kybernetiky

Studijní obor :	23 – 40 – 8 Automatizované systémy řízení ve strojírenství
Studijní zaměření :	automatizace inženýrských prací
Diplomant :	Petr Oliva
Téma práce :	laboratorní model řízení fotovoltaických článků
Theme of work :	Control of photovoltaic panels - laboratory model
Rok obhajoby DP :	2003
Vedoucí DP :	Ing. Slavomír Němeček
Konzultant :	Ing. Václav Lenk

Resumé :

Cílem diplomové práce je vytvořit laboratorní model řízení fotovoltaických článků. Řídící algoritmy jsou ovládány z PLC Simatic S7-300, který ovládá řídící jednotku krokového motoru FM 357-2. Řídící jednotka posílá synchronizační pulsy do výkonové jednoty krokového motoru. Výkonová jednotka již posílá řídící impulsy do krokového motoru. Celý systém je realizován bez zpětné vazby. PLC je spojeno s počítačem pomocí MPI kabelu.

Abstrakt :

The aim of my diploma thesis is to create control of photovoltaic panels – laboratory model. Control algorithms are controled by PLC Simatic S7-300 who control device FM 357-2. This device is used for sending control pulses to power unit of stepdrive. Power unit sending control power pulses directly to stepdrive. Whole system is maked without feedback. PCL is connected with computer with MPI cable.

Poděkování

Na tomto místě bych chtěl poděkovat Ing. Slavomíru Němečkovi a Ing. Václavu Lenkovi za odborné vedení, pomoc při zpracování diplomové práce, cenné rady a poskytnuté informace.

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č.121/2000 o právu autorském, zejména § 60 (školní dílo) a § 35 (o nevýdělečném užití díla k vnitřní potřebě školy).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

V Liberci 13. 5. 2003

…….

Petr Oliva

Místopřísežné prohlášení

„Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury.“

V Liberci 13. 5. 2003

…….
Petr Oliva

Obsah

1. Úvod

Solární panely jsou zdroje čisté elektrické energie, která je v posledních letech velice podporována. Všichni, kteří se zabývají zdrojem elektrické energie vědí, že zhruba od osmdesátých let se ve světě začínal razit trend čisté energie. Čistá energie je taková, která nedává jako vedlejší produkt škodlivé látky, jako například oxid uhličitý, který je vypouštěn do ovzduší z tepelných elektráren. Veliký problém byl tuto čistou energii dostatečně rozšířit, aby životní prostředí již nadále netrpělo neustálým znečišťováním. Jedním z takových řešení je použití netradičních zdrojů, jako například větrné elektrárny, vodní elektrárny nebo již zmiňovanou solární energii.

V této diplomové práci se budu zaměřovat výhradně na zdroj energie od slunečního záření. Existují dva druhy zařízení, které využívají solární energii v použitelné formě pro různé využití. Jedním z nich je solární panel sloužící na ohřev vody. Druhé zařízení slouží k přeměně sluneční energie na energii elektrickou. Tato zařízení se nazývají fotovoltaické články a po celém světě jsou hojně rozšířené. Používají se převážně na místech, která jsou celoročně osvícena slunečním zářením. Jejich účinnost není dostatečně vysoká, aby se dala běžně používat na kterémkoliv rodinném domku. K tomu by jich bylo zapotřebí značné množství.

V České republice je rozšířený právě první typ zařízení na ohřev vody. Mnozí lidé jej používají na předehřev vody v bojlerech a tím šetří elektrickou energii, které by byla nutná k samotnému ohřevu. Pouze přihřívají. Teplota vody se pohybuje kolem 60º C. Další hojně rozšířené použití je v předehřívání zahradních bazénů. V letních měsících je sluneční energie hojně po celý den a tak v čerstvě napuštěném bazénu ledovou vodou se lépe plave, pokud je předehřátá na požadovanou teplotu. Asi nejzajímavější řešení, se kterým jsem se osobně setkal je denní nabíjení akumulátoru, který slouží v nočních hodinách pro mírné osvětlení bytu.

V dalších části se budu zaměřovat pouze na fotovoltaické články, které mění světelnou energii na elektrickou energii. Ty jsou použity v laboratorní úloze, pro kterou v dalších částech navrhnu a realizuji řešení. Vlastní realizace je řešena programovatelným automatem PLC, který řídí všechny ostatní řídící či výkonové jednotky. Každé jednotce je věnována samostatná kapitola, kde je popsána její činnost, zapojení a případné propojení s ostatními jednotkami.

2. Fotovoltaické články

Vnitřní fotovoltaický jev umožňuje přímou přeměnu světelné energie na energii elektrickou. Tento jev byl objeven v roce 1839 Antoniem Besquelem. Poprvé byla využita v roce 1930 u fotonek a to s účinností pod 1%. V roce 1888 však objevil Hallwachs vnější fotoefekt. Díky nové technice polovodičů již existuje celá řada typů fotovoltaických článků. První fotovoltaický článek s účinností větší než 1% byl vyroben v roce 1954 v laboratořích Bell Telephone,USA. Měl účinnost 6%.

Typy používaných fotovoltaických článků :

- Mono-krystal Si (η=24,4%)
  Monokrystalický má perfektní stavbu, bez defektů a bez krystalických chyb
- Multi-krystalické SI (η=19,8%)
  Multi-krystalický má perfektní uspořádání jen v jednotlivých částech, asi 1 cm velkých zrnech.
- Amorfní Si (η=13,2%)
  Amorfní nemají uspořádané atomy a jsou nerovnoměrně rozloženy
- Vícevrstvové, tenkostěnné, s antireflexní vrstvou, s více PN přechody atd.

2.1. Fotoefekt

Fotoefekt rozdělujeme na tři základní :

- Chemický fotoefekt
- Vnější fotoefekt
- Vnitřní fotoefekt

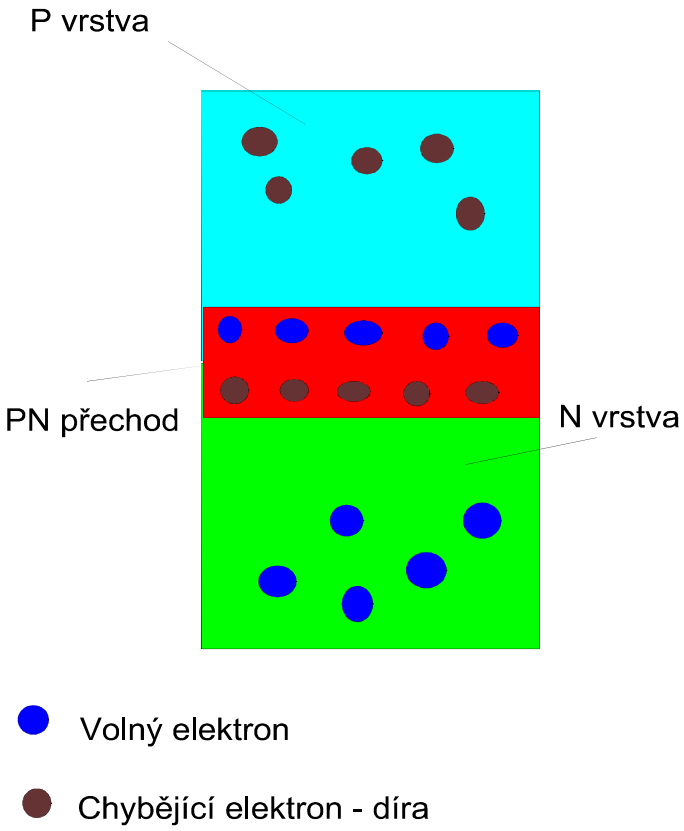
Chemický fotoefekt – například zčernání stříbrobromidové vrstvy při ozáření světlem. Fotoreakce nastává dělením stříbrobromidových molekul na atom stříbra a atom bromu.

Vnější fotoefekt – například uvolňování elektronů z povrchu kalia (kovu při ozáření světlem).Fotoreakce nastává přenosem fotonů s určitou energií na elektrony kalia.

Vnitřní fotoefekt – například zvýšením elektrické vodivosti nějakého polovodiče při ozáření světlem. Fotoreakce nastává přenosem fotonů s určitou energií na elektrony polovodiče

2.2. Struktura fotovoltaického článku

Elektrony se mnohou pohybovat pouze tehdy, jsou-li volné. Polovodiče jsou zvláštní materiály, které mají PN přechod. PN přechodem může elektron procházet od P k N a to pouze tehdy, má-li dostatečnou energii. Každý polovodič má dvě energetická pásma, ve kterých se může elektron udržovat. Valenční a vodivostní. Pokud není polovodič nějakým způsobem buzen (není mu dodána energie), elektron se udržuje ve valenčním pásmu. Pokud mu však dodáme energii, například pomocí slunečního záření (fotony o určité energii), elektrony excitují do vodivostního pásma. Tam vydrží nějakou dobu a zpátky rekombinují do valenčního pásma. Při absorpci předá foton svou energii elektronu, tím zvýší jeho energii a dovolí mu excitovat do vyšší vrstvy. Foton přitom zanikne a elektron se může volně pohybovat například v mřížce, kde je k dispozici procházejícímu proudu.



Obr. 2.1 – Struktura polovodiče

Nejpoužívanější materiál na výrobu polovodičů je křemík, na němž se vytvoří vrstva typu:

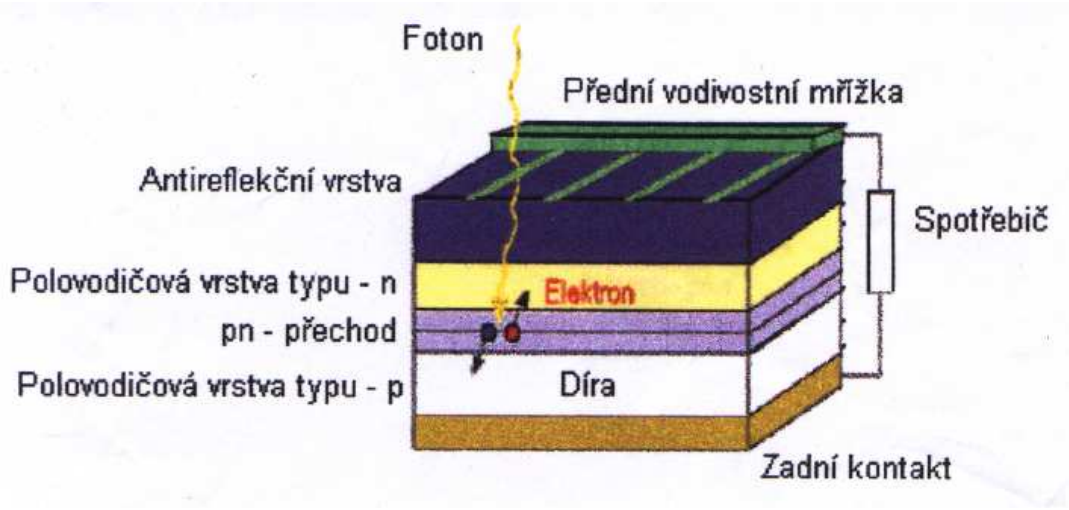
- N – například difuzí fosforu (zdroj volných elektronů je fosfor)
- P – například difuzí bóru (zdroj děr* je bór)

* díra = chybějící elektron ve valenční m pásmu

2.3. Princip fotovoltaických článků

Fotovoltaický článek je tvořen polovodičovým přechodem p-n, jehož jedna z částí je vystavována světelnému toku. Světlo bereme jako foton, který vnikl do p-n přechodu fotovoltaického článku a byl tam absorbován. Přechod p-n je tvořený oblastmi p a n položenými vedle sebe na stejném krystalu.

Ke zdroji světla je orientovaná vrstva n, jejíž tloušťka je většinou menší než 1µm. Pod touto vrstvou je základní materiál, p-vodivý křemík. Celková tloušťka fotovoltaického článku je kolem 300µm.



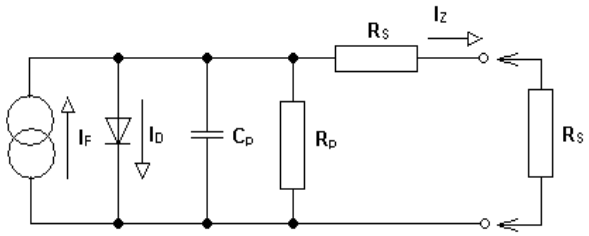
Obrázek 2.2 - Schéma fotovoltaického článku

Elektrony tedy přecházejí ke straně *n*-typu fotovoltaickéhočlánku a díry ke straně *p*-typu. Na přední straně fotovoltaického článku je záporný pól solárního generátoru proudu, na zadní straně je kladný pól. Tímto vynuceným přecházením je snížena pravděpodobnost rekombinace a tím pádem i ztráta elektronů cestujících k zápornému pólu. Celkový proudový okruh je uzavírán přes spotřebič a elektrony proudí zpět do fotovoltaického článku, jak je zobrazeno na obr.2.2. Slabou částí fotovoltaického panelu je kontakt na přední straně, tedy záporný pól. Z hlediska stínění by měl být co nejmenší, ale zároveň musí zaručovat elektrickou vodivost.

Poznámka :

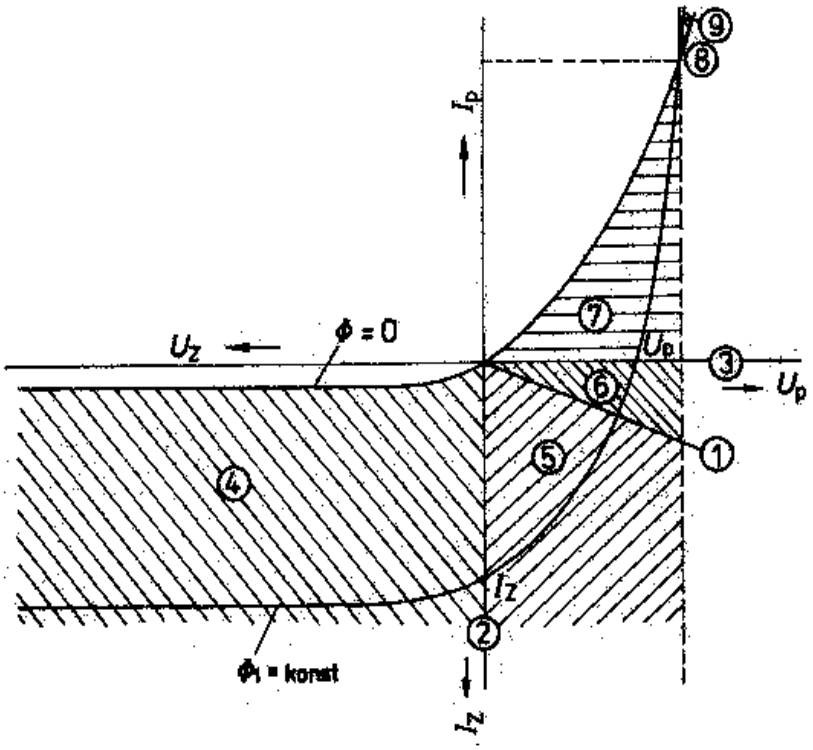
Proces výroby fotovoltaických článků je velice energeticky náročný. Pro výrobu fotovoltaického článku o jmenovitém výkonu 1kW je nutno vynaložit 1800kWh energie. Z toho je přibližně 50% na výrobu destičky a 50% na výrobu článku. Účinnost takového článku je poté kolem 35% a tak je nutné, aby článek pracoval alespoň 1,5 roku, aby se vrátili jen náklady na jeho výrobu.

2.4. Náhradní schéma fotovoltaického článku



If – proudový zdroj	Rs – odpor zátěže
Cp – parazitní kapacita	Io – proud naprázdno
Rp – parazitní odpor	Iz – proud zátěže

Obr. 2.3 - Náhradní schéma fotodiody



Obr. 2.4 - Voltampérová charakteristika fotodiody

Oblasti :

1 – Zatížení obecným odporem
2 – Snímá změny proudu s osvětlením
3 – Snímá změny napětí s osvětlením
4 – Závěrná oblast
5,6 – Pracovní oblast (hradlový efekt)
7 – propustná oblast
8 – Necitlivost na osvětlení

2.5. Veličiny fotovoltaického článku

Maximální výkon fotovoltaického článku (max. I a max. U):

$$P_{max} = U_{max} * I_{max} = U_0 * I_k * FF$$

U_{max} – maximální napětí při maximálním proudu

I_{max} – maximální proud při maximálním napětí

U_0 – napětí naprázdno

I_k – proud nakrátko

FF – faktor, určující kvalitu fotovoltaického článku. Vysoké hodnoty faktoru FF se dosáhne při dobré uzavírací schopnosti p-n přechodů, tj. při malých uzavíracích proudech diody, při malých sériových odporech a velkých paralelních odporech. Hodnoty FF s e pohybují v rozsahu 76 – 84 % a jsou funkcí teploty fotovoltaického článku.

2.6. Parametry solárního panelu

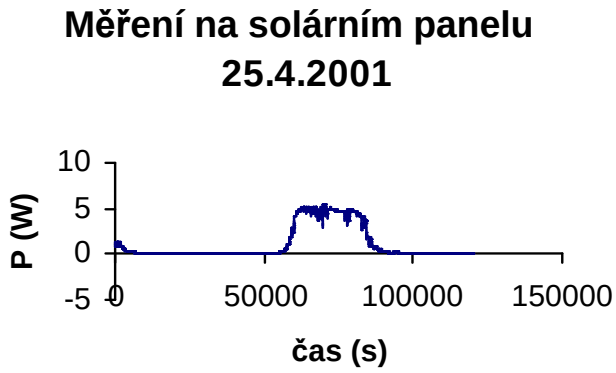
Použitý solární panel typu TGM 1500-12V má udávané parametry:

napětí U_{OC}	*: 18 V*
výkon (max)	*: 27 W*
proud (max) I_{SC}	*: 1500 mA*
proud (min) I_{SC}	*: 1400 mA*
rozměry článku (mm)	*: 51 x 102*
počet článků v serii	*: 36 kusů*
rozměry panelu (mm)	*: 533 x 447 x 25*
vestavěná dioda	*: ano*
vodiče vývodů s klipsy	*: ano*
váha	*: 3 kg*

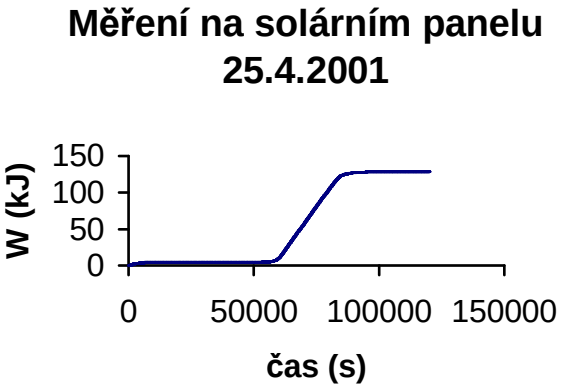
Solární panel s skládá ze 36 fotovoltaických článků, které jsou zapojeny do série. Tyto články jsou osvětlovány a dodávají na svém výstupu určité napětí, které se pohybuje kolem 10V. Toto napětí záleží velice na osvětlení. Čím větší osvětlení, tím větší napětí dostáváme na výstupu.

2.7. Měření na solárním panelu

Měření na solárním panelu bylo provedeno 25.4.2001. Podrobnější návod a postup [2].



Obr. 2.5 – Měření výkonu solárního panelu



Obr. 2.6 Měření práce solárního panelu

2.8. Sluneční záření

Slunce je obrovská žhavá plynná koule o hmotnosti $1989*10^{30}$ kg a průměru $1392*10^{6}$ km. Září díky tomu, že její povrch je zahřát na teplotu přibližně 6100K. Zářivost slunce je množství zářivé energie všech vlnových délek vyzářeným celým slunečním povrchem za 1 sekundu do celého prostoru, je to $3826*10^{26}$ W. Z toho dopadá za 1 sekundu na $1m^2$ plochy postavené kolmo k dopadajícím slunečním paprskům ve střední vzdálenosti Země od Slunce $1356W/m^2$. Tato energie je „motorem“ všech chemických, biologických a jiných procesů v atmosféře i na zemském povrchu.

Slunce vyzařuje energii v podobě elektromagnetického záření různých vlnových délek v rozsahu 100 – 10 000nm s maximem energie kolem 475nm. Sluneční elektromagnetické záření se dělí na ultrafialové, viditelné a infračervené.

Sluneční záření, které přichází k Zemi od slunečního disku a které je vzhledem k velké vzdálenosti Země od Slunce tvořeno svazkem rovnoběžných paprsků, nazýváme přímé sluneční záření. Při průchodu zemskou atmosférou je zeslabováno pohlcováním a rozptylem. Pohlcování atmosférou podlehne přibližně 15% přímého slunečního záření. To je způsobeno jednotlivými plynnými složkami atmosféry. Rozptyl je výsledkem odchylování paprsků na všechny strany od původního směru..

2.9. Východ a západ Slunce

Pro výpočet vzdálenosti Země od Slunce jsem použil aproximaci pohybu Země po elipse na pohyb po kružnici, čímž se výpočet zkreslil oproti vysílání „relace počasí“ max. o 1 minutu. Poloměr kružnice, po které se pohybuje Země kolem Slunce je R=150mil.km. Je to vlastně jedna astronomická jednotka. Poloměr Slunce R_S = 696 000 km a poloměr Země je R_Z = 6378 km. Pokud si představíme model Slunce – Země, pak můžeme vypočítat úhel, který vznikne, pokud spojíme tečnou Slunce a Zemi. Tento úhel je také úhel, o který se zvětší osvětlení Země od Slunce. Tento úhel nazvěme φ a jeho velikost je

$$\varphi = \arcsin\frac{R_s - R_z}{D} = 4.59 \cdot 10^{-3} \quad [^\circ]$$

Tento úhel je velice malý a proto jej v dalších výpočtech nepřipočítávám. Vznikne tedy chyba o velikosti 28 km při výpočtu délky osvětlené křivky, což je zanedbatelné.

Neboť je osa otáčení Země nakloněna o 23,5˚, není celá Země osvětlena v jednotnou hodinu stejně. Pro každou rovnoběžku je toto osvětlení různé. Když si představíme imaginární osu Země, která je kolmá a kolmo na skloněnou osu otáčení vytvoříme kolmici, vznikne nám část kružnice, která je osvětlená. Vztahuje se to k otáčení Země kolem Slunce. V jakou dobu bude osvětlena jaká část záleží na pootočení Země vůči Slunci. Nyní již potřebujeme určit úhel, který svírá část kružnice, která je osvětlená. Ten vypočítáme z Pythagorovy věty a úhel vyjde :

$$\varsigma = \arcsin(\tan\omega \cdot \tan\psi) \quad [^\circ]$$

Pokud víme, že jedno otočení Země je 2π a trvá 1440 minut (1 den), pak námi vypočítaný úhel ς vynásobený dvěma a odečtený od π podle trojčlenky přepočítáme na čas trvání osvětlení námi vypočítané části kružnice.

$$t = \frac{720}{\pi} \cdot (\pi - 2 \cdot \arcsin(\tan\omega \cdot \tan\psi)) \quad [\text{s}]$$

Dle zimního času je poledne rovno $\frac{t}{2}$.

Východ Slunce je tedy : $t_v = 720 \cdot \left[1 - \frac{1}{2\pi} \cdot (\pi - 2 \cdot \arcsin(\tan\omega \cdot \tan\psi))\right]$ [s]

Západ Slunce je tedy : $t_z = 720 \cdot \left[1 + \frac{1}{2\pi} \cdot (\pi - 2 \cdot \arcsin(\tan\omega \cdot \tan\psi))\right]$ [s]

ψ – zemská šířka (Praha ψ = 50˚)

ω - naklonění osy Země vůči slunci < -23.5 , +23.5> [˚]

ω = - 23.5 - je léto – nejdelší den (21.6.)

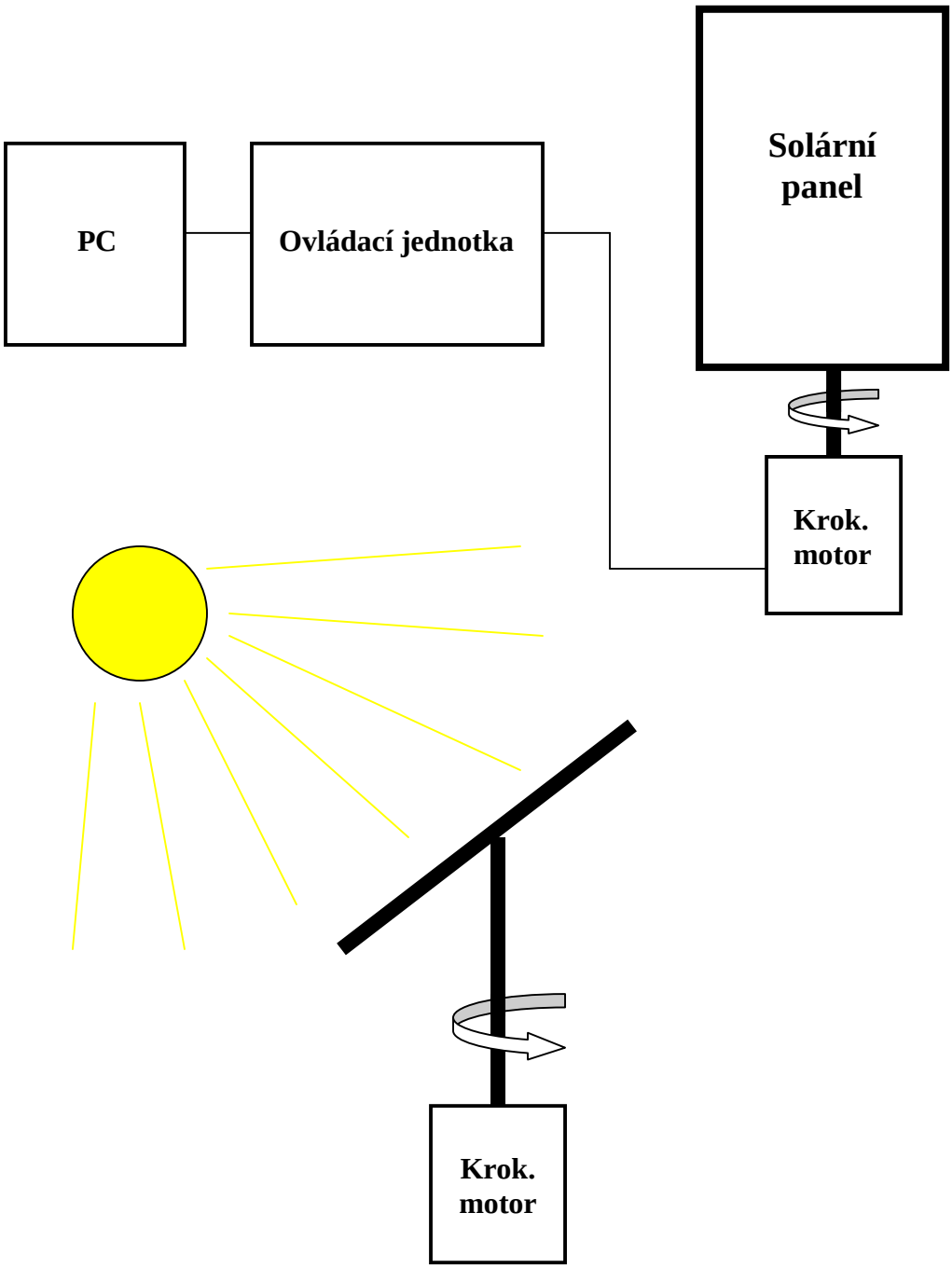
ω = 23.5 - je zima – nejkratší den (22.12.)

ω = 0 - jarní / podzimní rovnodennost

2.10. Nastavení solárního panelu

Protože při nastavení solárního panelu na určitou hodnotu sklonu nám vzniká chyba, při které přesně nesledujeme polohu slunce, nastal problém s nastavením solárního panelu na správnou polohu. Kdybychom řídili panel ve více osách než v jedné, tento problém by odpadl. Nejlepším řešením, jak tento problém alespoň minimalizovat, protože řešením pohybu pouze jedné osy se nedá zrušit, je nastavit sklon solárního panelu na hodnotu zemské šířky v daném místě. V České republice můžeme podle naší pozice nastavit sklon solárního panelu v rozmezí od 49^o do 51^o. V naší zeměpisné šířce jsem zvolil nastavení solárního panelu na sklon 50^o.

2.11. Mechanické řešení



2.12. Shrnutí

Požadavky na řídící systém :

- Pohonná jednotka musí umožňovat malé natočení
- Musí být schopná pracovat bez zpětné vazby
- Řídící jednotka musí být snadná na ovládání a nesmí být choulostivá na otřesy a na prašné prostředí
- Celý systém musí být alespoň částečně stíněný proti elektrickému rušení

Možné varianty řídícího systému:

1. a) Řídící jednotka je PC
   b) Ovládací modul je nutné navrhnout a postavit
   c) Pohonná jednotka je stejnosměrný motor

2. a) Řídící jednotka je PC
   b) Ovládací modul je nutné navrhnout a postavit
   c) Pohonná jednotka je krokový motor

3. a) Řídící jednotka je PLC
   b) Ovládací modul je možno objednat přímo k PLC
   c) Pohonná jednotka je stejnosměrný motor

4. a) Řídící jednotka je PLC
   b) Ovládací modul je možno objednat přímo k PLC
   c) Pohonná jednotka je krokový motor

Výběr nejvhodnějšího řídícího systému :

Po prostudování vlastností a požadavků na jednotlivé návrhy jsem došel k závěru, že PC je jako ovládací jednotka nevhodná, proto volím varianty 3 a 4, kde je řídící jednotka PLC. Tyto varianty jsou vhodné také proto, že k těmto PLC se dá dokoupit ovládací moduly a to jak pro krokový motor, tak při stejnosměrný motor. Zbývá tedy porovnat body 3 a 4. Výběr mezi stejnosměrným nebo krokovým motorem vyhrál krokový motor, neboť se jednoduše ovládá z PLC, není potřebná zpětná vazba a dokáže se natočit o velice malý úhel, což bylo jedním z požadavků. Proto volím variantu 4.

3. PLC

PLC je programovatelný logický automat, který je vlastně počítač do provozu. Je oproti obyčejným počítačům odolný vůči některým vlivům, kterým by počítač v krátké době podlehl. Jedná se o vibrace, prach a vlhkost.

Pro ovládání solárního panelu jsme si vybrali PLC od německé firmy Siemens. Zvolili jsme PLC S7-300 s CPU 314IMF. Ale v neposlední řadě digitálními vstupy a výstupy, které pracují s napěťovými úrovněmi 0V a 24V. Máme k dispozici 16 digitálních vstupů a 16 digitálních výstupů. Každý vstup má svou adresu, které jsou 124.x a 125.x pro vstupy, kde x označuje bit. Každý výstup má také svou adresu. Ta je kupodivu stejná se vstupy a je označena 124.x a 125.x Rozdíl k jejich přístupu se řeší až programově a to bude ukázáno později.

PLC je spojeno s počítačem přes kartu také od firmy Siemens, která s PLC komunikuje po sériovém kabelu pomocí protokolu RS 485 (MPI kabel). Pomocí tohoto rozhraní lze připojit i ovládací panel.

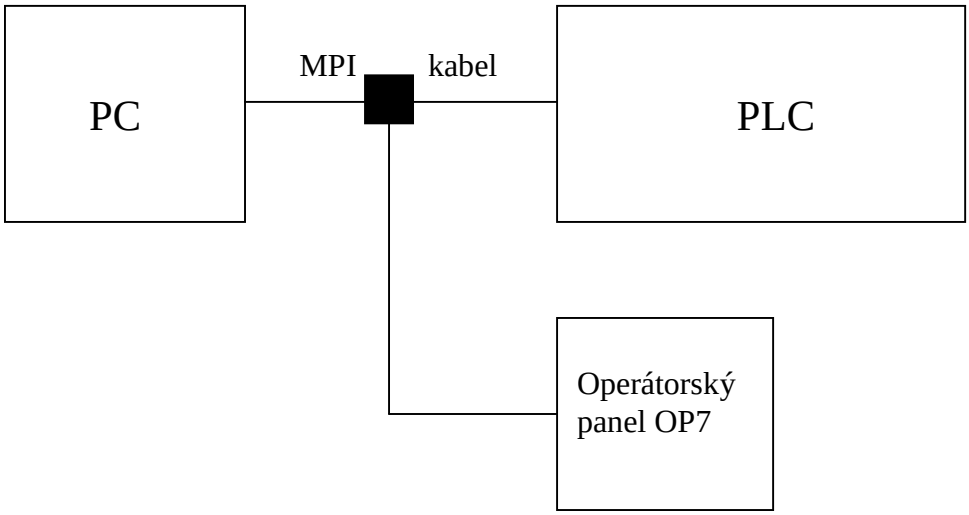
3.1. Zapojení

Dříve, než se budeme zabývat zapojením PLC, musíme poukázat na velice důležitou věc a to je, že všechny moduly počínaje PLC a konče FM StepDrive používají pro své ovládání 24V. Firma Siemens pro tuto příležitost nabízí napájecí modul PS300. Tento modul je připojen na síťové napětí 230V a má dva nízko úrovňové napěťové výstupy na 24V, které jsou označovány L+ a dva výstupy na uzemnění označené jako M. Těchto 24V bude použito jak na napájení PLC, tak i pro napájení modulu FM 357-2. Maximální proudový odběr je 2A, což při řešení této úlohy určitě nedosáhneme.

Teď již zpátky k dalšímu zapojení. Zapojení PLC S7-300 je velice jednoduché. Po nainstalování PG/PC katry do počítače jsme přes vytvořený sériový port RS 485 vytvořili spojení PC s PLC. PLC připojím na bezpečné napájecí napětí 24V z PS300. Předposledním a nejdůležitějším krokem je nainstalování ovládacího klíčku, bez kterého bychom později PLC nemohli vůbec ovládat. Poslední věc potřebná k instalaci je zakomponování Lithiové baterie, která bude v pozdější době udržovat ovládací program v paměti. Celá instalace se provádí velice jednoduše, vše je umístěno na jednom místě na PLC zepředu a je to šikovně

překryto otevíracím panelem. Veškeré kontakty jsou vytvořené svorkovnicí, takže připojení jakéhokoliv dalšího vodiče je velice jednoduché.

V této sekci by bylo hodno zmínit se také o svislé řadě diod, které jsou umístěny na přední straně PLC v levém horním rohu. První dioda je označena jako SF a pokud se vyskytne jakákoliv softwarová nebo hardwarová chyba, tak se rozsvítí červeně. Druhá dioda je označena BAT a pokud není baterie správně nainstalována, tak svítí červeně. Pokud není baterie nainstalovaná, PLC bude fungovat normálně, pouze bude tato dioda svítit. Třetí dioda indikuje, že PLC běží a že je připraveno. Je označena DC5V a svítí zeleně. Čtvrtá dioda je FRCE a pátá dioda je RUN a signalizuje uživateli, že program, který spustil běží, nebo se spouští. Pokud dioda bliká, tak se program spouští a když přestane blikat a již jen svítí, tak program běží normálně. Pokud při spouštění programu nastane chyba, rozsvítí se chybová dioda SF. Poslední dioda je STOP a pokud program stojí, tak svítí žlutě.Jinak nesvítí.



Obr. 3.1 - Propojení PC / PLC / panel OP7

3.2. Ovládání

Ovládání PLC je dvojí. První je klíčkem umístěným na přední straně panelu. Tento klíček může být přepnut do několika pozic. První a standardní pozice je STOP. V této pozici může být do PLC nahráván program z počítače a PLC je ve stavu *„zastaveno“*. Druhá pozice je RUN. V této pozici je PLC spuštěn, probíhá nahraný program, ale nemůže do něj být v průběhu nahrán žádný další program. Třetí pozice je RUN-P. Má stejnou

funkce jako RUN, avšak můžeme nahrávat program z počítače,aniž bychom jej zastavovali. Tuto pozici jsem velice využíval při programování. Poslední pozice je MRES a je nestálá, neboť při přepnutí do ní musíte klíček v této poloze držet. V této pozici je možno PLC resetovat podržením klíčku alespoň 3 sec v této poloze. Dioda STOP je rozbliká a poté již jen svítí. Vrátíme klíček do původní polohy a hned přepneme znovu do polohy MRES a držíme. Dioda STOP se rozbliká rychleji a po 3 sec. opět již svítí. Tím je vymazáno všechno z PLC. Pokud se při vykonávání programu nebo při instalaci vyskytne nějaká chyba, rozsvítí se indikační dioda na přední straně (SF – svítí červeně).
Druhé ovládání je pomocí programu, který je nahrán do PLC po sériové lince, která je označena MPI (multi – point – interface). Pomocí MPI lze připojit k PC/PG kartě nejen PLC, ale i ovládací panel. Programové vybavení a samotný programovací jazyk bude popsán podrobně v kapitole 3.3.

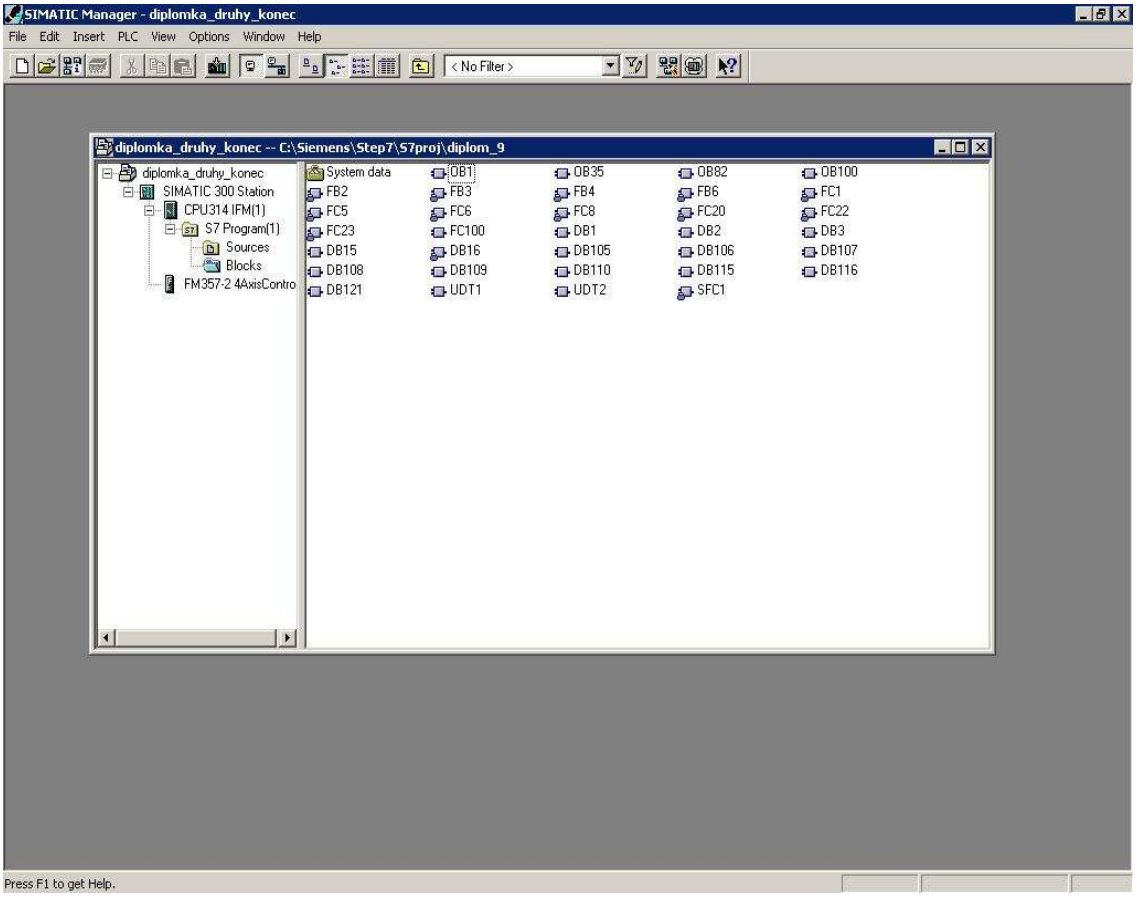
Obr. 3.2 –PLC SIMATIC S7-300 s CPU314 IMF

3.3. Programové vybavení

K PLC S7-300 škola zakoupila programovací software SIMATIC STEP 7. Je to nejnovější programovací software od firmy Siemens. Tento program se po nainstalování musí autorizovat z přiložené diskety. Po spuštění programu se objeví okno, kde si vybereme, jaký druh PLC máme k dispozici. Po vybrání PLC se zobrazí další tabulka, kde je výběr, jaké bloky se mají připravit pro programování. OB1, OB2 atd. Každý z těchto bloků má svou vlastní funkci. OB1 se cyklicky opakuje, po spuštění PLC se vykonává pořád dokola. Další funkční bloky jako např. OB35 (cyklické přerušení) se vykonávají buď po jejich zavolání, nebo pokud je splněna podmínka, která jejich start způsobí. Po pečlivém zvolení všech potřebných bloků stačí již nechat dokončit „*wizarda*“ počáteční nastavení a již se nám zobrazí programovací prostředí.

3.3.1. Programovací prostředí

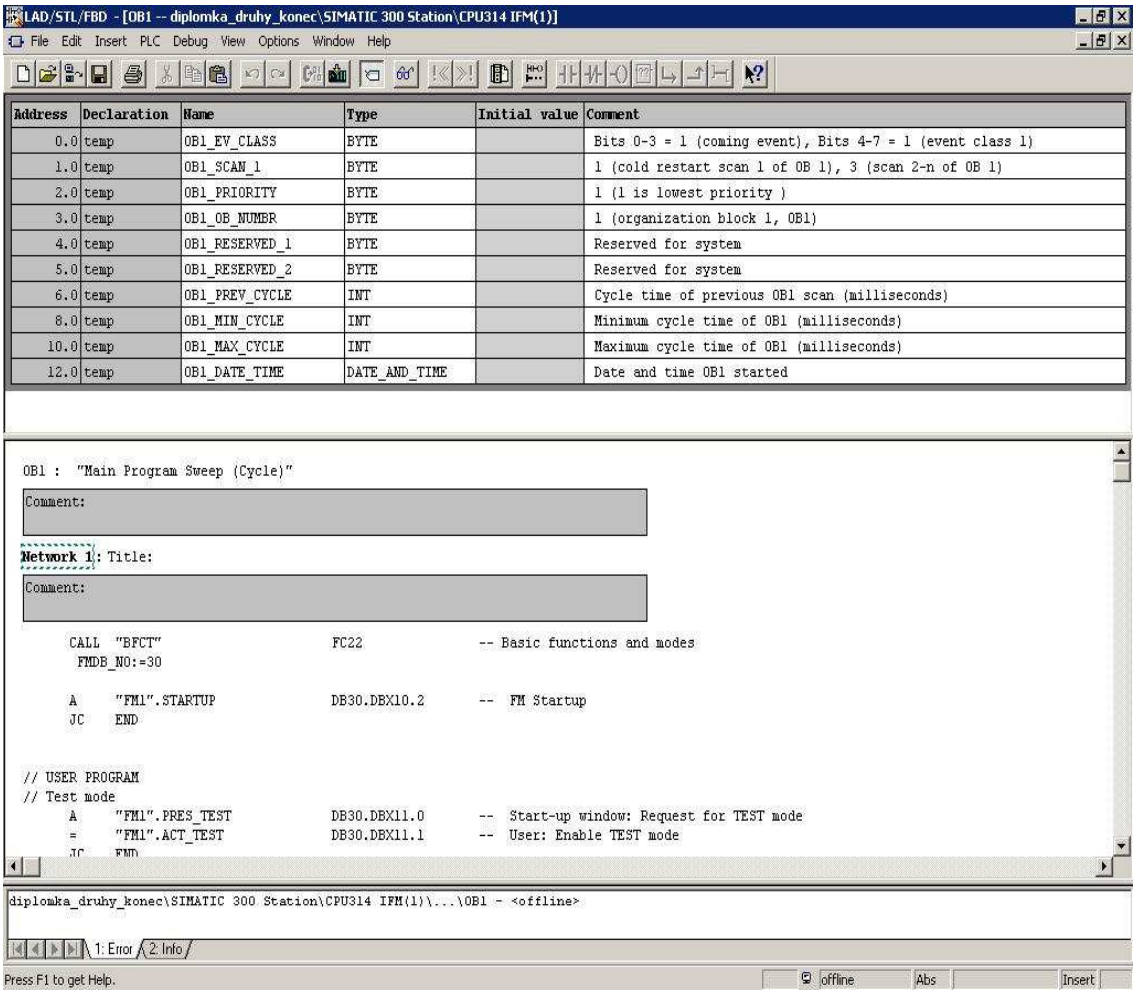
Programovací prostředí je rozděleno do několika částí. V první části, která se nám objeví po spuštění, je přehledně zobrazeno, jaká zařízení jsou připojena k PLC a jaký druh PLC máme. Pod každým zařízením je složka S7-program, kterou když rozbalíme, máme na výběr dvě záložky. První je *Sources* a druhá je *Blocks*. V *Sources* mohou být uloženy námi předdefinované a popsané proměnné. V *Blocks* jsou uloženy všechny bloky, které jsme v počátku zvolili, nebo postupně přidali. Pokud poklepeme na jakýkoliv blok, který není chráněn zámkem (pouze bloky před programované od Siemens), otevře se nám STL/LAD/FBD okno, ve kterém se tyto bloky již mohou programovat. Máme na výběr tři druhy programování. Každý z nich umožňuje všechny funkce jako ostatní, záleží pouze na stylu programování a jeho zobrazení. První je styl LAD. Při použití tohoto stylu používáme připravené bloky, které máme na výběr v horní liště. Každý blok má své vstupy a výstupy, na které se připojují různé další. Tento způsob je velice podobný programování v Matlab – Simulink.



Obr 3.1 – Programovací prostředí STEP 7

Druhý styl je STL. Při použití tohoto stylu je nutné znát alespoň základní syntax příkazů, neboť programování probíhá zápisem příkazů podobně jako v assembleru. Třetí styl programování je FBD. Je velice podobný LAD programování, používají se také připravené bloky, ale vizualizace je mnohem přehlednější než u LAD. Všechny tyto styly se dají navzájem kombinovat. Jediná podmínka je, aby každá sekce byla programována pouze v jednom z nich.

Jak jsem se již zmínil, programování je rozděleno do sekcí. Každá z nich je nazvána *Network* a číslo. V každé sekci je možno různým stylem naprogramovat různé věci. Počínaje posláním log.1 na výstup nebo její přečtení až po práci s časovači nebo čítači. Podrobnější popis všech použitých příkazů bude následovat vždy při jejich použití.



Obr 3.2 – LAD/STL/FBD programovací prostředí

3.3.2. Popis příkazů

Pokud chceme začít programovat, je nutné se seznámit s běžně používanými příkazy. Hned na začátku upozorňuji, že příkazy budu vysvětlovat a zobrazovat pouze v LAD stylu programování, protože je podle mého velice přehledný. Každý blok, pokud ho použijeme, či je jen označen, má v programu zabudovaný *HELP*. V tomto helpu je anglicky/německy popsána stručně činnost jednotlivých bloků a také co bloky očekávají na vstupu / výstupu (datové typy). Právě s datovými typy jsem měl při programování největší problémy. V STEP 7 existuje mnoho datových typů, ale zde jsem použil pouze některé z nich.

Použité datové typy :

Typ	Přednastavená hodnota
INT	0
DINT	L#0
DWORD	DW#16#0
BOOL	FALSE
S5TIME	S5#T0s
DATE_TIME	L#0
DATE	L#0
TIME	L#0

Než začneme se samotným popisem, musím upozornit, že ve STEP 7 je možno programovat v anglickém stylu nebo v německém stylu. V dalších odstavcích budu popisovat pouze anglický styl. Popisovat budu, jako u datových typů, pouze příkazy použité. Ostatní je v případě potřeby nutno dohledat v dokumentaci.

První a jeden z nejčastěji používaných příkazů je zápis na výstupy a čtení ze vstupů. Jak jsem se již zmínil v části 3, vstupy a výstupy jsou označené 124.x a 125.x a pouze záleží na příkazu, jestli to je vystup či výstup. Jako příklad budu používat adresu 124.0 pro vstup a pro výstup 125.0.

Výstup : *I 124.0* Vstup : *Q 125.0*

Tento příkaz sám osobě nic neudělá, jen nastaví, zda-li se jedná o vstup či výstup. Pokud chceme s těmito adresami něco učinit, musím použít příkaz *SET / RESET*.

SET *I 124.0*

nastaví požadovaný výstup do log. 1

RESET *I 124.0*

nastaví požadovaný výstup do log. 0

Dalším použitým blokem je ***COMPARATOR***. Ve STEP 7 existuje mnoho druhů komparátorů. Pro různé datové typy (*INT / DINT / WORD / DWORD*) a také jakým způsobem vyhodnocují vstupní data.

Existuje pět typů :

- je rovno
- je větší než
- je menší než
- je menší nebo rovno než
- je větší nebo rovno než

Pokud je daná podmínka splněna na výstupu, který je vždy označen Q se objeví 1, bloky se chovají, jako by byly zdrojem napětí pro následující blok. Tímto je zaručeno, že pokud není nějaký blok splněn a druhů ano, tak na výstupu není nic.

Následující blok slouží pouze k zápisu určité hodnoty do proměnné. Je označen ***MOVE*** a má jeden vstup a jeden výstup. Na vstup přiřadíme hodnotu, která má být uložena do proměnné na výstupu. Hodnoty u všech bloků mohou být předány přímo, nebo pomocí proměnné.

Po nich bude následovat soubor příkazů, které mají co dočinění s časem a datumem. V PLC existují hodiny reálného času, které, pokud je nainstalovaná baterie, udržují čas a datum. Tento čas a datum se dá přečíst pomocí bloku ***READ_CLK***. Tento blok po zavolání načte čas a datum z PLC a uloží ho na výstup. Na výstupu tedy musí být proměnná typu *DATE_TIME*, jinak blok bude svítit červeně. Takto svítí každý blok, kterému něco chybí, nebo je připojena proměnná se špatným typem.

Příkaz ***DT_DATE*** umožňuje ze zabaleného typu *DATE_TIME*, který obsahuje jak čas, tak i datum, dostat zvlášť datum. Na výstupu musí být proměnná typu *DATE*. Na vstupu tohoto bloku musí být proměnná, ve které je uložen čas a datum, ve formátu *DATE_TIME*.

Příkaz ***DT_TIME*** umožňuje ze zabaleného type *DATE_TIME* dostat zvlášť čas. Na výstupu musí být proměnná typu *TIME* a na vstupu proměnná typu *DATE_TIME*.

Další řada použitých příkazů je aritmetického typu. Jedná se o ***DIV***, ***MUL***, ***SUB*** a ***ADD*** (dělení, násobení, odečítání, sčítání). Každý z těchto bloků opět existuje v několika variantách a to jak pro reálná čísla, tak i pro celá. Pro celá čísla jsou ještě rozdělena na WORD, INT a jejich rozšíření . Všechny mají dva vstupy a jeden výstup. První vstup

znamená „*co*“ a druhý „*s čím*“. Na výstupu je vždy požadovaný výsledek ve tvaru, který jsme zvolili použitým blokem.

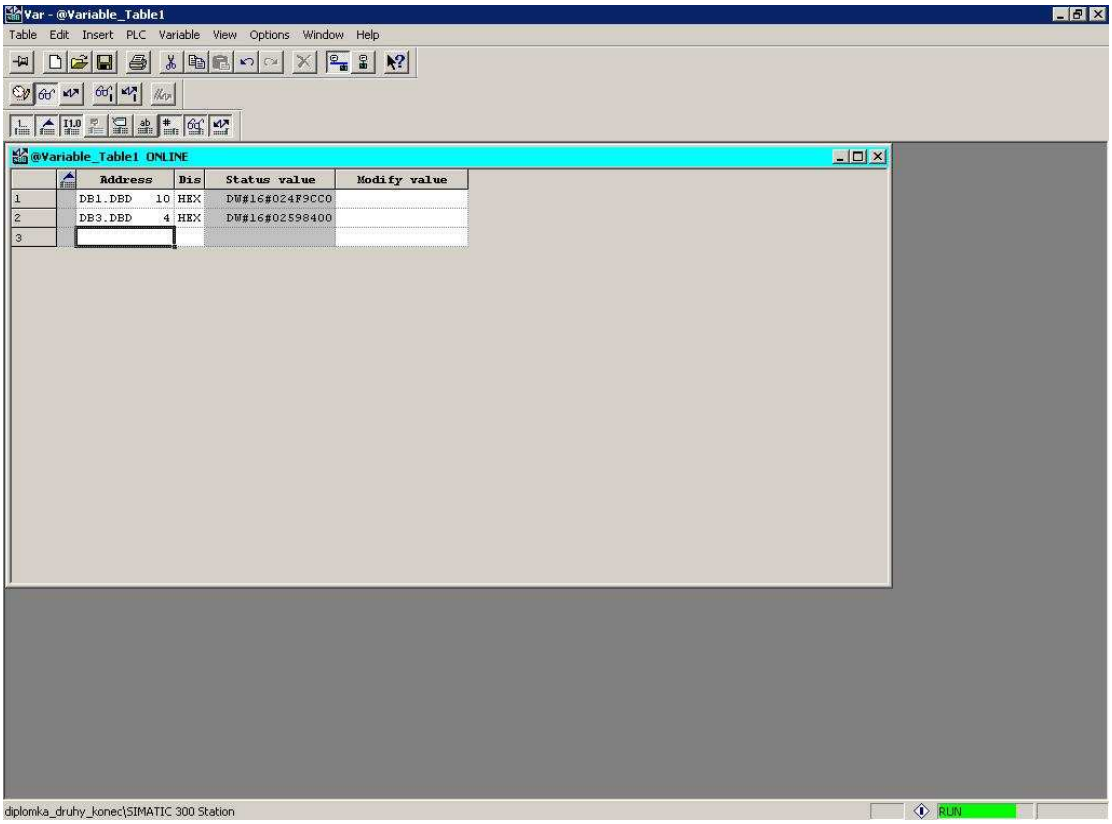
Posledním používaným blokem je ***TIMER*** neboli časovač. Těchto bloků existuje mnoho druhů.

- s odpočítáváním dolů
- s přičítáním nahoru
- s přičítáním nebo odčítáním (dle nastavení)

Zde jsme použili s odpočítáváním dolů. Tento blok má čtyři vstupy a jeden výstup, Ošetřené vstupy musí být však jenom dva, ostatní jsou nepovinné. První vstup slouží k odstartování časovače. Pokud je na vstupu log. 1, tak začne *TIMER* pracovat (v našem případě odčítat směrem dolů). Druhý vstup je nepovinný a je to *SET*. Pokud je zde přivedena log. 1 , tak je časovač nastaven na počáteční hodnotu, která je uložena ve třetím povinném vstupu. Tam je hodnota ve tvaru *S5 TIME*. V jiném tvaru není hodnota akceptována. Poslední vstup je *RESET* a pokud je na tento vstup přivedena log. 1, tak je celý *TIMER* nastaven do počátečních hodnot a opět čeká na startovací impuls. Výstup z časovače je Q a pokud je odpočítávání ukončeno, tak se na tomto výstupu objeví log. 1. Pokud není odečítání ukončeno, nebo není aktivován, je tam log. 0.

Ostatní příkazy jsem nepoužíval a tak není nutné se zde o nich zmiňovat. Pouze ještě jednou odkazuji na výborný *HELP*. Příkazy, které slouží ke komunikaci s modulem FM 357-2 budou popsány v kapitole 4.2.

Když jsme se již seznámili s příkazy a datovými typy, je vhodné se zde zmínit, že zde existuje také „***Monitor proměnných***“. Je přístupný z menu a po jeho výběru se objeví nové okno, ve kterém máme volná pole (jako v excelu), do kterých vyplníme název proměnné a pokud PLC běží a přepnuli jsme se do „*on-line*“ módu (ikona brýle), tak se zobrazí aktuální hodnoty. Tyto hodnoty pak můžeme při běhu programu měnit. Toto okno je velmi užitečné při ladění programu (obr 3.3). Program můžeme sledovat za běhu také v LAD/STL/FBD prostředí, kde se přepneme do „*on-line*“ módu (ikona brýle). Vedle příkazů se vytvoří okno, ve kterém jsou aktuální hodnoty všech příkazů a hodnot v programu. Je to trochu nepřehledné, ale pokud potřebujeme sledovat několik příkazů najednou, tak se to hodí.



Obr. 3.3 – Monitor proměnných

3.3.3. Nahrání programu do PLC

Pokud jsme již ze samotným programováním hotovi, musí se tento připravený program dostat z PC do PLC. K tomu slouží nainstalovaná karta v PC a MPI kabel, který propojuje PC s PLC.

PLC musí být zapnutý (napájení 24V musí být připojeno). Zapnutý stav je indikován zelenou diodou DC5V. Dále máme dvě možnosti, jak mít přepnutý klíček v PLC. Jedním je STOP a druhů je RUN-P. V obou těchto pozicích může být do PLC nahrán program z PC.

V programu STEP 7 se samotné nahrání „*download*“ realizuje pomocí ikony v horním panelu. Je tam vyobrazení PLC a žlutá šipka směřující do něj. Tato ikona je aktivní pouze v případě, že máme označenu záložku nazvanou „*blocks*“ nebo v STL/LAD/FBD. Pokud jsme na záložce „*blocks*“, tak se nahraje blok, který je označen, nebo pokud není označen žádný, tak se nahrají všechny. Pokud jsem v STL/LAD/FBD, tak je nahrán pouze blok, který mám právě otevřený. Pokud již daný blok v PLC existuje, tak je po odsouhlasení přepsán novým.

3.4. Spojení s FM 357-2

Samotná realizace spojení PLC s FM 357-2 je pomocí vnější sběrnice. Není nutné žádné speciální zapojení. Firma Siemens dodává k modulu právě již zmíněnou vnější sběrnici. Je to malá černá krabička, která složí pouze k propojování. Vzadu na PLC a FM 357-2 jsou kontakty, na které se tato sběrnice připojí. Komunikaci mezi moduly ovládá PLC a to samostatně. Siemens nikde neuvádí, jakým způsobem je komunikace prováděna. Velikou výhodou je, že se již toto propojení nemusí programovat. Po připojení PLC a FM 357-2 k napájení je provedena komunikace a nastavení FM 357-2 do stavu „*připraven*“.

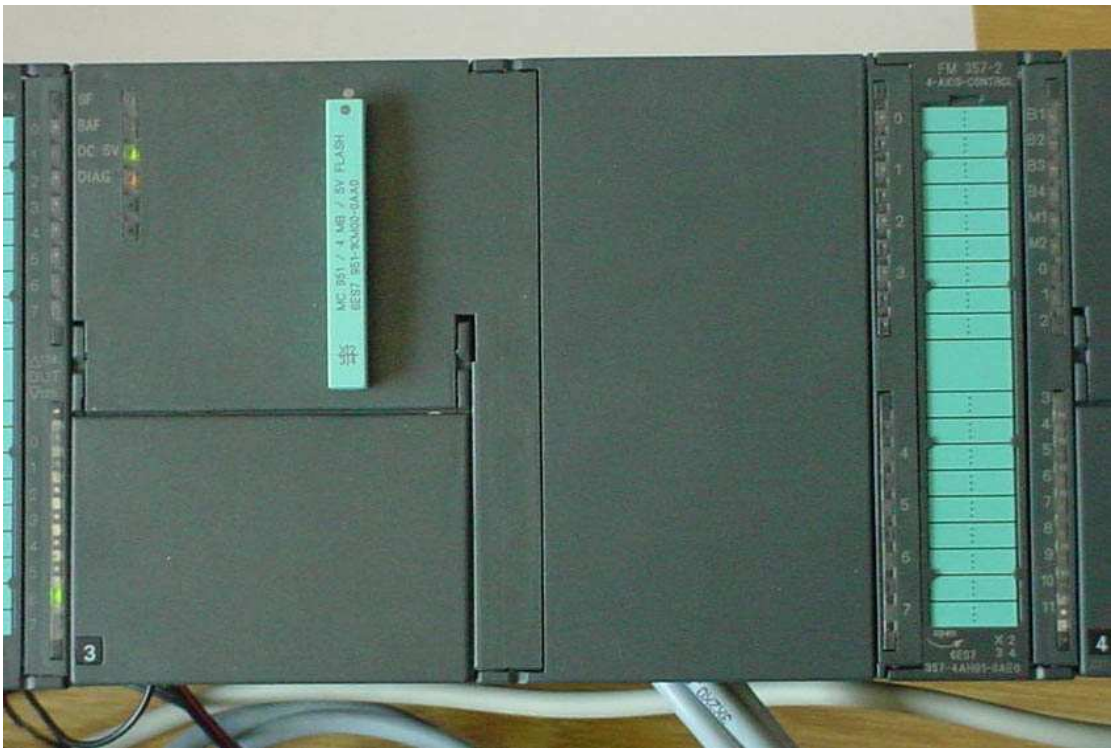
4. FM 357-2

Modul FM 357-2 je postavený přímo pro připojení k PLC S7 – 300. Je napájen jako PLC 24V DC. Slouží jako ovládací modul pro výkonový člen krokového motoru. Dokáže ovládat až čtyři osy, zde však používáme pouze jednu, která je označena jako AXIS 1. Komunikace s PLC je pomocí vnější sběrnice, jak již bylo zmíněno v předchozí kapitole. FM 357-2 má jeden výstupní konektor označený jako sub - D (50 pinů - X2) a čtyři vstupy označené jako sub – D (15 pinů – ENCODER). Modul obsahuje „*paměťovou kartu*“, na které je autorizační software a také volné místo, na které se může nahrát program.

4.1. Zapojení

Zapojení modulu FM 357-2 není vůbec těžké. Napájení je z napájecího modulu a to 24V DC . Před samotným napájením je však bezpodmínečně nutné, aby byla nainstalována „*paměťová karta*“. Karta je pouze zasunuta. Po jejím zasunutí je možné připojit napájení. Z této karty je nutné přenést autorizační firmware do modulu FM 357-2 jinak nebude fungovat. Přenesení tohoto firmware je detailně popsáno ve firemní dokumentaci, přímo v souboru „*fm357-2.pdf*“ v kapitole 3.2 – *Installing firmware / firmware update*. Pokud je dodržen přesný postup, je vše hotovo v pár minutách. Po nainstalování firmwaru je nutné ponechat kartu v modulu, protože po každém zapnutí je tento firmware kontrolován. Pokud firmware nesouhlasí, či je špatně nainstalován, je to indikováno diodami na přední straně panelu.

První dioda je SF a pokud existuje jakýkoliv softwarový, či hardwarový problém buď svítí, nebo bliká červeně. Druhá dioda je BATF a ta svítí červeně pouze tehdy, je-li špatně nainstalovaná baterie, či úplně chybí. Třetí dioda je DV5V a ta zeleně signalizuje připojení modulu k napájení. Poslední diodou je DIAG. Ta bliká žlutě, pokud je firmware ověřený a modul je připraven k práci. Pokud nastane jakákoliv chyba, diody blikají v různé kombinaci. Tyto kombinace a návrh řešení problémů je detailně popsáno ve firemní dokumentaci v souboru „*fm357-2.pdf*“. Pokud jsme nainstalovali firmware a FM 357-2 je připojen k PLC, můžeme již přejít k programování tohoto modulu přes STEP 7.



Obr 3.3. – FM 357-2 s paměťovou kartou

4.2. Ovládání

Samotné ovládáni je řešeno bloky, které se musí vložit do samotného programu ve STEP 7. Blok, který jsem použil je BFCT a slouží k nastavení základních funkcí a operací s FM 357-2. Jako parametr má číslo datového bloku, který bude používat pro komunikaci s programátorem pomocí proměnných. V tomto datovém bloku je uloženo veliké množství proměnných, které jednak ovládají motor a také jej monitorují. Bližší vysvětlení všech použitých proměnných bude provedeno v kapitole 7. Opět však nebudou popisovány všechny proměnné, ale pouze použité.

4.3. Spojení s FM StepDrive

Spojení FM 357-2 s ovládacím modulem FM StepDrive je realizováno pomocí propojovacího kabelu, který má na jedné straně sub – D (X2) a na druhé straně má čtyři výstupy, pro každou osu jeden. Tyto výstupy jsou zapojeny do FM StepDrive a pulsy ovládají krokový motor. Pro každou použitou osu musí být samostatný modul FM StepDrive.

Popis sub – D (50 pinů) :

pin	jméno	pin	jméno	pin	jméno
1	SW1	*18*	ENABLE1	*35*	SW2
2	BS2	*19*	ENABLE1_N	*36*	BS3
3	SW3	*20*	ENABLE2	*37*	SW4
4	BS4	*21*	ENABLE2_N	*38*	PULSE1_N
5	PULSE 1	*22*	GND	*39*	DIR1_N
6	DIR1	*23*	GND	*40*	PULSE2
7	PULSE2_N	*24*	GND	*41*	DIR2
8	DIR2_N	*25*	GND	*42*	PULSE3_N
9	PULSE3	*26*	ENABLE3	*43*	DIR3_N
10	DIR3	*27*	ENABLE3_N	*44*	PULSE4
11	PULSE4_N	*28*	ENABLE4	*45*	DIR
12	DIR4_N	*29*	ENABLE4_N	*46*	nepoužit
13	nepoužit	*30*	nepoužit	*47*	RF1.2
14	RF1.1	*31*	nepoužit	*48*	RF2.2
15	RF2.1	*32*	nepoužit	*49*	RF3.2
16	RF3.1	*33*	nepoužit	*50*	RF4.2
17	RF4.1	*34*	BS1		

Popis signálů pro krokové motory :

PULSE[1..4], PULSE[1..4]_N	Hodinové pulsy a negace
DIR[1..4], DIR[1..4]_N	Signál směru a negace
ENABLE[1..4], ENABLE[1..4]_N	Povolení serva a negace
GND	Signálová zem

Ostatní signály nejsou pro krokový motor, ale pro stejnosměrné motory, proto jejich význam zde neuvádím.

5. FM StepDrive

FM StepDrive je výkonová jednotka, která ovládá krokový motor posíláním sledu impulsů do jeho cívek. Tento modul je ovládán z FM 357-2, o kterém jsem si již zmínil. Tuto komunikaci provádí bez zásahu čehokoliv jiného. Programované pulsy se v těchto modulech samy programují. Jediné, co mohu ovlivnit, jsou napěťové vstupy na tomto modulu, kterými povoluji, nebo zakazuji komunikaci s motorem. Tyto vstupy a výstupy budou popsány v následující kapitole.

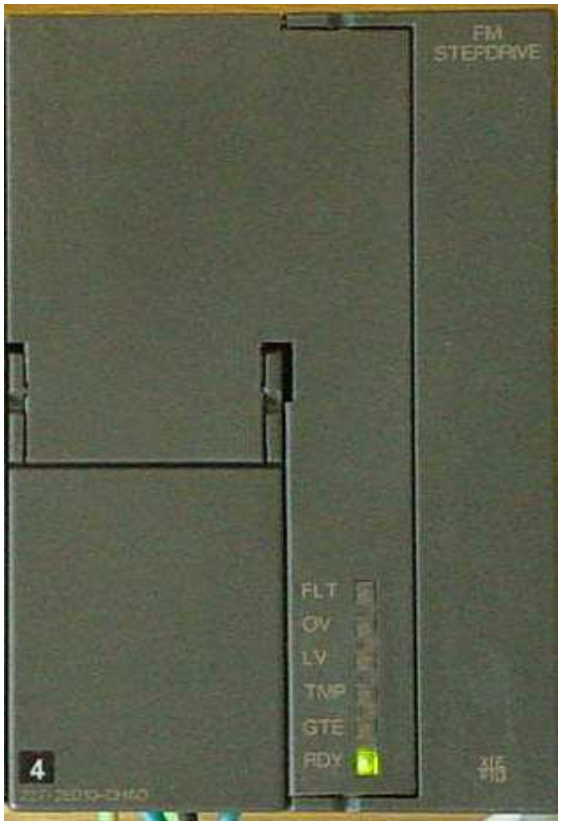
5.1. Zapojení

Zapojení tohoto modulu je již složitější než u všech ostatních bloků. Jedním z důvodů je to, že musí mít vlastní zdroj 230V AC, který slouží k napájení pulsů pro krokový motor. Dále musíme z napájecího modulu vzít 24V DC a napájet vstupy a výstupy na tomto modulu. Tyto vstupy a výstupy slouží ke komunikaci s programátorem, který přes ně může celý systém otáčení efektivně řídit, aniž by musel zasahovat do řídících impulsů v FM 357-2.
Popis vstupů/výstupů modulu :
Vstupy :
GATE_N – pokud je na tomto vstupu přítomno 24V DC, PULSE jsou povoleny
ENABLE_N – pokud je na tomto vstupu přítomno 24V DC, motor je napájen
Výstupy :
ZERO – pokud je tam přítomno 24V DC, motor se otáčí
READY2 – pokud je tam přítomno 24V DC, modul je aktivní a připraven
MSTILL – pokud je na GATE_N 0V, je na MSTILL 24V
Podrobnější popis vstupů a výstupu je popsáno ve firemní dokumentaci v souboru „fm_fbst.pdf“ v kapitole *4. Signal description*.

5.2. Spojení s krokovým motorem

Spojení s krokovým motorem je realizováno pomocí výkonového stíněného kabelu. Tento kabel má tři vodiče, které jsou připojeny na výstupy FM StepDrive U,V,W a na stejné vstupy u krokového motoru U,V,W. První izolační vrstva je zároveň zapojována jako zemnící vodič. Druhá izolace slouží pouze k odstínění pulsů ve vodiči.

Výstupy / vstupy U,V,W jsou kvalitně popsány na bloku a na motoru. Tímto správným zapojením je spojení realizováno. Grafický popis zapojení lze nalézt ve firemní dokumentaci v souboru „fm_fbst.pdf“ v kapitole *6. Wiring.*



Obr 3.4. - StepDrive

6. Krokový motor

Krokový motor patří do jiné, zvláštní skupiny motorů. Jeho pohyb je realizován po krocích, které jsou vykonávány při vzájemném působení cívek v motoru. Každá z cívek je nějakým způsobem napájena a jejich vzájemnou kombinací je zaručeno, že se motor otáčí. Krokové motory mají velikou výhodu v přesnosti polohování a jejich velkou nevýhodou je možnost přetížení. Pokud krokový motor přetížíme, tak může dojít ke ztrátě kroku a tím ke znehodnocení celého otáčení.

Náš krokový motor je zapojený k výkonovému čelenu FM StepDrive pomocí výkonového vodiče. Tento výkonový vodič má dvě stínící vrstvy. První je zároveň použita jako zemnící vodič pro krokový motor. Druhá pouze stíní. Ve výkonovém kabelu jsou tři vodiče, která slouží k přenášení impulsů do krokového motoru. Tyto vodiče se zapojí v motoru na vstupy U,V,W a stínící vrstva jako zemnění.

Toto jednoduché spojení již zajistí, že po povolení otáčení a přivedení pulsů do FM StepDrive se motor začne otáčet přesně podle našeho přání.

Technické parametry krokového motoru :

Rozsah pracovních teplot : 0 – 40 ° C

Maximální frekvence pulsů : 4,3 kHz

Maximální počet otáček : 6000 min^{-1}

Maximální moment : 2 Nm

Maximální proud : 1,75 A

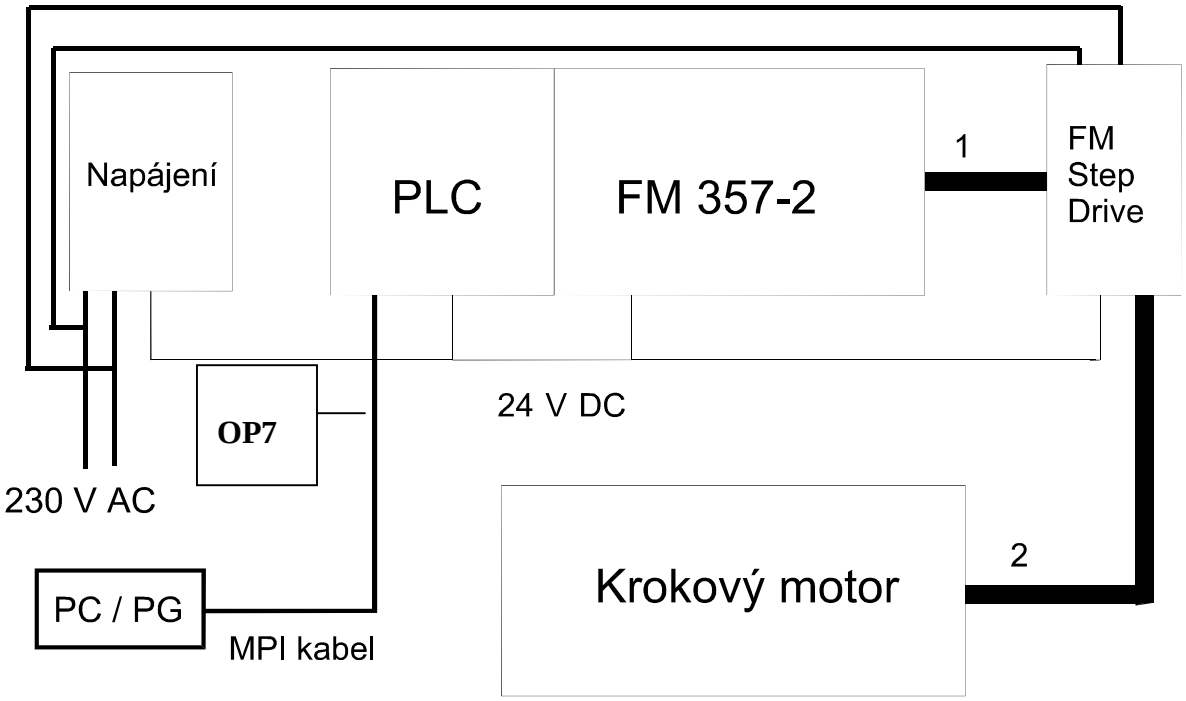
Moment setrvačnosti rotoru : 1,1 $kgcm^2$

7. Zapojení řídícího obvodu

Řídící obvod pro ovládání krokového motoru je realizován následujícím systémem. Řídící jednotkou tohoto řetězce je programovatelný automat PLC SIMATIC S7-300. V tomto modulu je umístěn ovládací program, který kontroluje, řídí a počítá vše potřebné, co je nutné pro správné řízení otáčení krokového motoru a tudíž i pro správné nastavení solárního panelu vzhledem k slunci. PLC je vnější sběrnicí připojen k ovládacímu modulu pro krokový motor FM 357-2, který je řízen programem právě z PLC. Tento program nastavuje v určité kombinaci proměnné pro FM 357-2, ty mu jsou vnější propojovací sběrnicí poslány a modul samotný je již spravuje a nastaví správné signály na výstupní konektor X2. Tento konektor je pomocí signálového kabelu spojen s výkonovým modulem FM StepDrive, který již poslané signály z FM 357-2 vyhodnotí a nastaví příslušné hodnoty na výstupech U,V,W. Tyto výstupy slouží jako přímé vstupy řídících signálů pro krokový motor. V klidovém stavu, tedy pokud je modul FM StepDrive pouze napájen a neposílá žádné pulsy do krokového motoru, jsem naměřil na těchto výstupech 98V. Při posílání pulsů se mi nepodařilo změřit napětí na těchto vstupech, proto jej zde neuvádím. Výstupy z modulu FM StepDrive jsou zapojeny již přímo na vstup do krokového motoru a to na vstupy U,V,W.

7.1. Hardwarové spojení

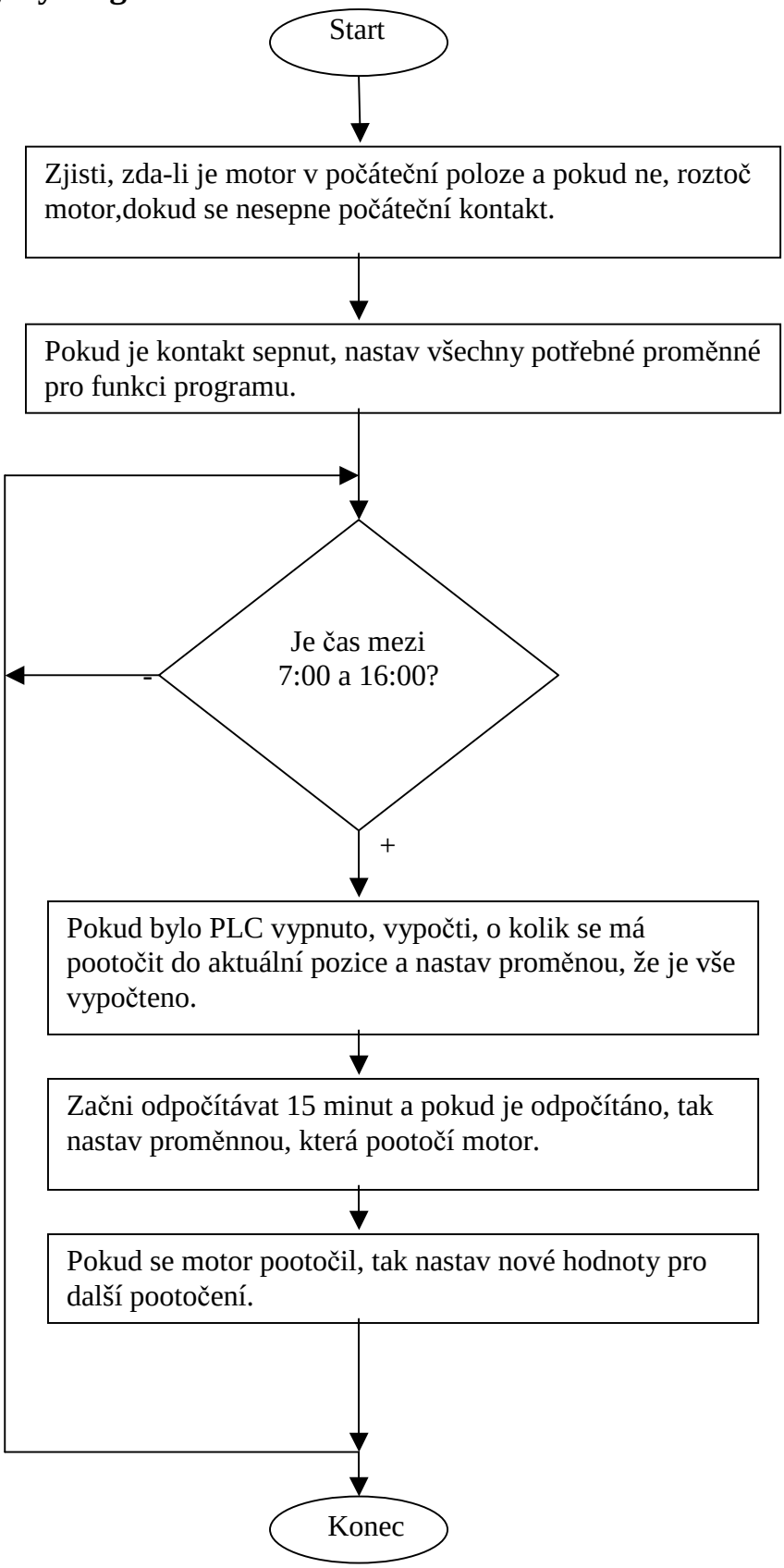
Následující obrázek nám graficky znázorňuje, jak je daný systém zapojen. Pokud jsou dva moduly vedle sebe bez naznačeného propojení, předpokládáme, že jsou zezadu propojeny vnější sběrnicí. Ostatní propojovací kabely budou označeny a popsány, aby nedošlo k jejich záměně. Každý modul bude popsán stejným způsobem, jakým jsem se o něm zmiňoval v předcházejících kapitolách.



1 - propojovací kabel FM 357-2 a FM StepDrive
2 - výkonový kabel k propojení FM StepDrive a krokového motoru

Obr. 7.1 – zapojení řídícího obvodu

7.2. Vývojový diagram



7.3. Program

V kapitole 3.3.2 jsem již popsal základní příkazy, které se vyskytují v tomto programu. Záměrně jsem se předtím nezmínil o použití Booleovské algebry, která je zde zastoupena v podobě příkazů *AND, OR, NAND, NOR* a dalších. Těmito příkazy testuji přítomnost napětí na vstupech do PLC a také všechny proměnné typu *BOOL*. V programu je mnohokrát použit přístup k proměnným a ke každé se přistupuje specifickým způsobem. Jedním z nich je zavolání čísla datového bloku (např. *DB1*), za ním zapsat tečku (.) a pak název naší proměnné. Pokud daný datablok(*DB*) existuje a proměnná je nadeklarována, tak se její jméno znemí v interní označení. Například *DB1.pomocna_bool* se změní na *DB1.DBX0.0*. Proměnná *pomocna_bool* je typu *BOOL*. Druhý způsob přístupu k proměnným je zapsání přímo typu proměnné a její polohu v datové oblasti. Například *DB1.DBD10*. Každá proměnná má své specifické označení za *DB*. Toto označení bude patrné z programu, který bude v dalších kapitolách rozebrán podrobně. Následující kapitoly jsem rozdělil podle toho, jak jsem postupoval při tvorbě programu a jak je samotný program vykonáván. Program bude v celé jeho délce uveden na konci v příloze. Zde bude i s originálním anglickým popisem od výrobce. Zde se budu snažit tento popis realizovat v češtině a to co nejsrozumitelněji.

V programu je použito mnoho bloků, která slouží pro komunikaci s FM 357-2, jako datové bloky mnou vytvořené, či vytvořené bloky pro ovládání motoru. Také jsou zde bloky FCxxx, ve kterých může být část programu, který používáme. Zde je použit blok FC100, ve kterém inicializuji proměnné pro FM 357-2, které ovládají pulsy pro krokový motor. Tyto proměnné budou vysvětleny v kapitole 7.2.13.

7.3.1. Příprava prvotního programu

Pro přípravu prvotního programu pro ovládání krokového motoru jsem po dlouhém přemýšlení použil část příkladu, který firma Siemens dodala k softwaru. Do OB1 jsem zkopíroval spojení s FM 357-2. K tomuto modulu je však nutné zkopírovat datové a ostatní bloky, které jsou s tímto blokem spojeny. „USERDB“ a „DB30“ bloky musely být také nakopírovány do programu. Tímto jsem zaručil samotnou komunikaci mezi bloky PLC a FM 357-2, aniž by se v další době objevila chyba. Takto připravenou kostru programu jsem pak již systematicky doplňoval různými dalšími bloky a programovým textem za účelem roztočení motoru dle časového schématu. V následujících kapitolách jsou jednotlivé části podrobně popsány.

7.3.2. Spojení s FM 357-2 na programové úrovni

Spojení PLC a FM 357-2 je realizováno jedním blokem FBCT. K tomuto bloku je potřeba přidat ještě jeden datový, ve kterém budou uloženy informace o stavu FM 357-2. Jako vstupní parametr do tohoto bloku je právě číslo datového bloku.

```
CALL  "BFCT"              // Základní funkce a operační módy FM 357-2
 FMDB_NO:=30              // Číslo datového bloku pro FM357-2 signály

A     "FM1".STARTUP       // Start první FM 357-2
JC    END                 // Dokud není nastaven, tak žádný uživatelský
                             program nesmí být vykonáván
```

A "FM1".STARTUP znamená, že pokud není první FM 357-2 (max. počet 4) připraven (není dokončena komunikace), tak se nesmí vykonávat žádný další program v PLC. Návěští *END* je umístěno na úplném konci programu.

```
// Zde začíná uživatelský program
A     "FM1".CONNECT       // Je připraven pro komunikaci
JCN   END                 // Jestliže ne, skoč na END
// nastavení Override
L     B#16#26             // Nastavení 100 %
T     "FM1".CH[1].CTR.FD_OVERR
T     "AX1".AX[1].CTR.OVERR
S     "FM1".CH[1].CTR.FD_OVERR_FDR
S     "AX1".AX[1].CTR.ACT_OVERR   // Aktivuj

CALL  FC100    // zavolá pomocný blok FC100, ve kterém jsou uloženy další
                  nastavení pro FM 357-2. Tyto proměnné jsou vysvětleny
                  v kapitole 7.2.14.
```

7.3.3. Kontrola pozice motoru

Pozice motoru je kontrolována hned po spuštění programu. Správnou polohu v počátku indikuje sepnutý kontakt, který je připojen na vstup 125.7. Pokud tento kontakt nebyl na počátku sepnut, tak celý program stojí a motor se otáčí tak dlouho v mínus směru, dokud se daný kontakt nesepne. Pokud je sepnut, tak se toto sepnutí uloží do proměnné DB2.DBW0 číslo 1. Pokud kontakt nebyl nebo ještě není sepnutý, tak je v této proměnné uložena 0.

```
    AN   I   125.7                    // není sepnut počáteční kontakt
    A(                                 // a ještě nebyl sepnut tak
    L    DB2.DBW   0
    L    0
    ==I
    )
    S    "USERDB".DIR_M_AX1    // otáčej motorem v mínus směru

// pokud byl sepnut počáteční kontakt, tak ulož 1 do DB2.DBW0
    A    I   125.7                     // byl již sepnut ?
    JNB  oli                           // pokud ne, tak skoč na návěští oli
    L    1                             // jinak ulož do BD2.DW0 1
    T    DB2.DBW   0
    SET
    SAVE
    CLR
oli: A    BR                           // pokud vše proběhlo v pořádku, tak
    R    "USERDB".DIR_M_AX1    // přestaň otáčet motorem
```

7.3.4. Práce s časem a datumem

Pomocí následujících třech bloků jsem realizoval čtení aktuálního času a datumu z PLC. Použil jsem tři proměnné pro jejich uložení. První je *DB1.cas_datum*, ve které je uložen čas i datum dohromady ve zvláštním formátu. Druhá je *DB1.DBW14* do které jsem uložil již samotné datum z *DB1.cas_datum*. A poslední použitou proměnnou je *DB1.DBD10*, do které jsem uložil aktuální čas z DB1.cas_datum.

```
// přečti hodiny a datum
    CALL  "READ_CLK"              // volání bloku pro čtení času a datumu
     RET_VAL:=DB1.DBW8
     CDT   :=DB1.cas_datum        // načtené hodnoty ulož do DB1.cas_datum
    NOP   0
// přečti datum zvlášť
    CALL  "DT_DATE"               // volání bloku pro čtení datumu
    IN    :=DB1.cas_datum         // vyextrahuj datum z proměnné DB1.cas_datum
```

```
      RET_VAL:=DB1.DBW14       // a ulož jí do proměnné DB1.DBW14
      NOP   0
// zjisti čas zvlášť
      CALL  "DT_TOD"             // volání bloku pro čtení času
       IN    :=DB1.cas_datum     // vyextrahuj čas z proměnné DB1.cas_datum
       RET_VAL:=DB1.DBD10        // a ulož jí do proměnné DB1.DBD10
      NOP   0
```

7.3.5. Povolení ENABLE_N a GATE_N

Pokud jsou splněny podmínky :

- byl motor v počáteční poloze (*DB2.DBW0 = 1*)
- je čas větší než 7:00 (*L#25200000)*
- je čas menší než 16:00 (*L#57600000*)

tak na výstupy 125.0 a 125.2 pošli log.1. Tyto výstupy jsou spojeny s ENABLE_N a GATE_N na FM StepDrive a ovládají povolení otáčení motoru ve správný čas. Zaručují, že se motor nebude otáčet v noci, ale jen přes den.

```
// pokud je správný čas, tak povol ENABLE_N a GATE_N
      A(                          // je v proměnné DB2.DBW0 jednička?
      L     DB2.DBW    0
      L     1
      ==I
      )                           // pokud ANO, tak
      A(                          // pokud je v proměnné DB1.DBD10 větší hodnota
      L     DB1.DBD   10          // než 25200000 tak,
      L     L#25200000
      >=D
      )
      A(                          // pokud je v DB1.DBD10 menší hodnota než
      L     DB1.DBD   10          // 57600000 , tak
      L     L#57600000
      <D
      )
      =     Q    125.0            // tak nastav 1 na výstupy 125.0 a
```

```
=    Q    125.2              // 125.2
```

7.3.6. Kontrola vypnutí PLC

V některých okamžicích je nutné PLC vypnout a ponechat ho v tomto stavu nějakou dobu. Pokud v této době by se měl motor pootočit za sluncem, nastane problém, protože pokud je PLC vypnuté, nic se otáčet nemůže. Čas v PLC však běží dál a nehledí na naše potřeby. Proto jsem vytvořil tuto část programu, která zaručí, že po znovuzapnutí PLC se vrátí do startovací polohy a poté je znovu nastaven na tu správnou. Tato část programu pouze nastavuje potřebné proměnné pro tuto realizaci. Samotné natočení je realizováno v jiných částech programu, které jsou však těmito proměnnými ovládány.

Abychom poznali, že PLC bylo vypnuto, používám výstup 125.6, na kterém je po zapnutí a nastavení do správné polohy 1. Pokud vypnu PLC, tak je tam 0. Tento výstup je spojen se vstupem 125.6, který tyto hodnoty monitoruje a dle toho rozpozná, že má nastavit motor do nové pozice.

```
    A(
    A(
    AN   I   125.6            // vstup, na kterém je 0, pokud se PLC vypnulo
    JNB  ol1
    L    0
    T    DB2.DBW   0          // do proměnné uchovávající informaci, že motor byl
    SET                       // v počátku nastav 0, aby se motor natočil zpět na počátek
    SAVE
    CLR
ol1: A    BR
    )
    JNB  ol2
    L    DW#16#18E4120
    T    DB3.DBD   4          // do proměnné uchovávající čas startu PLC ulož
    SET                       // hodnotu 7:00
    SAVE
    CLR
ol2: A    BR
```

```
)
JNB   ol3                  // do proměnné uchovávající čas konce PLC ulož
L     DW#16#18E4130  // hodnotu 16:00
T     DB3.DBD   16
SET
SAVE
CLR
ol3:  A     BR
S     Q    125.6          // na výstup 125.6 nastav 1
R     DB2.DBX    2.1    // nastav do proměnné DB2.DBX2.1 hodnotu“ false“
```

7.3.7. Pootočení motorem

Pokud byl sepnut kontakt počátku a čas je po rozmezí, které je dáno proměnnými *DB3.DBD4* a *DB3.DBD16* tak vydá pokyn začni otáčet motorem v plus směru. Časovač zařídí, že se motor bude otáčet jen 20ms, což je dostatečně dlouho, aby se natočil o potřebný úhel.

```
//Pokud je sepnut počáteční kontakt, tak začni otáčet motorem.Musí být i správný čas.
A(                              // byl motor v počátku ?
L     DB2.DBW    0
L     1
==I
)
A(                              // je čas větší než první rozmezí?
L     DB1.DBD   10
L     DB3.DBD    4
>D
)
A(                              // je čas menší než druhé rozmezí?
L     DB1.DBD   10
L     DB3.DBD   16
<D
)
S     "USERDB".DIR_P_AX1  // roztoč motor
```

//Až se rozjede motor, tak čítej 20ms a pak motor vypni.

```
A    "AX1".AX[1].CTR.DIR_P     // pokud se točí motor
A    DB2.DBX    2.1            // a není to nastavení do poč. polohy
AN   DB2.DBX   18.0
L    S5T#20MS                  // tak nastav čas 20ms
SD   T     1                   // do časovače T1
A    T     1                   // a až dojde na nulu, tak
R    "USERDB".DIR_P_AX1        //přestane točit motorem
R    T     1                   // vyresetuje časovač
S    DB2.DBX    2.0            //a  nastaví DB2.DBX2.0 na true
```

7.3.8. Nastavení nových hodnot pro motor

Pokud jsem otočil motorem jako natočení panelu po 15 minutách, tak se nastaví proměnná *DB2.DBX2.0* na *true* a tak mohu přičíst do porovnávacích časových proměnných 15 minut. To zařídí, že se motor pootočí znovu za 15 minut.

// už jsem otočil motorem a tak přičti 15 minut a pak otoč znova

```
A(
A    DB2.DBX    2.0           // pokud je otočeno, tak
JNB  oli1
L    DB3.DBD    4            // přičti do proměnné 15min
L    DB3.DBD   12
+D
T    DB3.DBD    4
AN   OV
SAVE
CLR
oli1: A    BR
)
JNB  oli2
L    DB3.DBD   16            // a do téhle proměnné taky
L    DB3.DBD   12
+D
```

```
    T    DB3.DBD   16
    AN   OV
    SAVE
    CLR
oli2: A    BR
    R    DB2.DBX   2.0          // zruš informaci o otočení
```

7.3.9. Nastavení hodnot pro další den

Pokud se program dostane do situace, že čas je menší než 7:00 nebo větší než 16:00, tak se nastaví všechny používané proměnné na počáteční hodnoty, tedy na hodnoty odpovídající 7:00. Vynuluje se i proměnná, která udržuje informaci o tom, že motor byl v počáteční poloze.

```
// Až budou čtyři hodiny, tak vynuluj počáteční kontakt, nastav zpátky hodnoty pro otáčení
další den.
    A(
    A(
    A(
    L    DB1.DBD   10              // pokud je méně jak 7:00
    L    L#57600500
    >=D
    )
    A(
    L    DB1.DBD   10              // nebo více jak 16:00, tak
    L    L#57601000
    <=D
    )
    JNB  oli3
    L    0
    T    DB2.DBW   0               // nastav proměnnou počáteční kontakt na 0
    SET
    SAVE
    CLR
oli3: A    BR
```

```
    )
    JNB   oli4
    L     DW#16#18E4120         // proměnnou na porovnání času nastav na 7:00
    T     DB3.DBD    4
    SET
    SAVE
    CLR
oli4: A     BR
    )
    JNB   oli5
    L     DW#16#18E4135         // a druhou na 16:00
    T     DB3.DBD   16
    SET
    SAVE
    CLR
oli5: A     BR
    R     DB2.DBX    2.1         // vyresetuj pomocné proměnné
    R     DB2.DBX   18.1
```

7.3.10. Správné hodnoty pro otáčení po spuštění

Tento kód se vykoná hned po spuštění programu, otestuje, zda-li byl motor v počáteční poloze, přečte aktuální čas a vypočítá rozdíl, který je od počáteční polohy k poloze, ve které by měl motor být. Dle toho nastaví časové porovnávací proměnné a vypočítá, jak dlouho se má motor otáčet, aby toto časové zpoždění dohnal a nastavil se do správné polohy.

```
// nastaví hodnoty otáčení na správné hned po spuštění
    A(
    A(
    A(
    A(
    AN    DB2.DBX    2.1         // ještě kód neproběhl?
    A     I    125.6
    A(
    L     DB2.DBW    0          // byl motor v počáteční poloze?
```

```
    L     1
    ==I
    )
    A(
    L     DB1.DBD   10          // je čas větší než 7:00
    L     L#25200000
    >=D
    )
    A(
    L     DB1.DBD   10          // a menší než 16:00
    L     L#57600500
    <D
    )
    JNB   oli6
    L     DB1.DBD   10       //odečti aktuální čas od 7:00
    L     L#25200000
    -D
    T     DB2.DBD    4      // a ulož do časové porovnávací proměnné 1
    AN    OV
    SAVE
    CLR
oli6: A     BR
    )
    JNB   oli7
    L     DB2.DBD    4       // vyděl ho 15 minutami
    L     L#900000
    /D
    T     DB2.DBD    4      // a ulož do stejné proměnné
    AN    OV
    SAVE
    CLR
oli7: A     BR
    )
    JNB   oli8
```

```
    L     DB2.DBD    4      // zpět ho vynásob 15 minutami
    L     L#900000
    *D
    T     DB2.DBD    4
    AN    OV
    SAVE
    CLR
oli8: A     BR
    )
    JNB   oli9
    L     DB2.DBD    4     // a toto číslo přičti ke každé časové proměnné
    L     DB3.DBD    4
    +D
    T     DB3.DBD    4
    AN    OV
    SAVE
    CLR
oli9: A     BR
    )
    JNB   oliv
    L     DB2.DBD    4
    L     DB3.DBD   16     // a tady ke druhé
    +D
    T     DB3.DBD   16
    AN    OV
    SAVE
    CLR
oliv: A     BR
    S     DB2.DBX   12.0   // zde nastav, že vše bylo vykonáno
    S     DB2.DBX    2.1   // nastavením true do proměnných
```

7.3.11. Výpočet času na první otáčení po startu

Zde je počítáno, jak dlouho se musí motor po zapnutí otáčet, aby se dostal do aktuální pozice, která by odpovídala pozici, kdyby byl zapnut ještě před sedmou hodinou a do této pozice by se dostal sám.

```
// vypočítej, jaký je vymezený čas na otáčení po startu
    A(
    A(
    A(
    A(
    A    DB2.DBX   12.0        // mohu to vypočítat?
    JNB  o1
    L    DB1.DBD   10          // uložím si aktuální čas
    T    DB3.DBD   20
    SET
    SAVE
    CLR
o1: A    BR
    )
    JNB  o2
    L    DB3.DBD   20          // uložím jej i do další proměnné
    T    DB3.DBD   24
    SET
    SAVE
    CLR
o2: A    BR
    )
    JNB  o3
    L    DB2.DBD    4         // pak jej vydělím 15 minutami
    L    L#900000
    /D
    T    DB2.DBD    8        // uložím do pomocné proměnné
    AN   OV
    SAVE
```

```
    CLR
o3: A   BR
    )
    JNB  o4
    L    DB2.DBD   8          // vynásobím jej 20ms pro získání času
    L    L#20
    *D
    T    DB2.DBD   8          // uložím do pomocné proměnné
    AN   OV
    SAVE
    CLR
o4: A   BR
    )
    JNB  o5
    L    DB3.DBD   24          // a přičtu jej k proměnné do které jsem si uložil čas
    L    DB2.DBD   8
    +D
    T    DB3.DBD   24
    R    DB2.DBX   12.0        //  nastavím pomocné proměnné
    S    DB2.DBX   18.0        // abych mohl roztočit motor na tuto vypočtenou dobu
o5: NOP  0
```

7.3.12. Natočení do správné pozice po startu

Tato část programu využívá vypočtené hodnoty na otočení motoru do správné polohy. Otestuje nejdříve, jestli jsem tyto hodnoty již spočítal a mohu je tedy použít a pokud ano, tak motorem otáčím tak dlouho, jak jsem již předtím spočítal.

```
// otoč motorem do pozice, která je ta správná
    A    DB2.DBX   18.0          // je vše již spočítáno a mohu otáčet?
    A(
    L    DB1.DBD   10            // mohu v tomto čase? (< 7:00 )
    L    DB3.DBD   20
    >=D
    )
    A(
```

```
L    DB1.DBD   10              // a v tomto taky? (> 16:00)
L    DB3.DBD   24
<=D
)
S    "USERDB".DIR_P_AX1   // tak začni otáčet v plus směru

// až to bude otočeno, tak to vyrezetuj a je to

A    DB2.DBX   18.0        // pokud otáčím
A(
L    DB1.DBD   10          // a čas na to vymezený skončil, tak
L    DB3.DBD   24
>D
)
R    "USERDB".DIR_P_AX1    // přestaň otáčet motorem
R    DB2.DBX   18.0              // a nastav výpočet na 0 = provedený
```

7.3.13. Nastavení proměnných pro motor v bloku FC100

V bloku FC100 jsou nastaveny všechny proměnné, které slouží k nastavení pohybu motoru, či jeho monitorování. Každá proměnná je popsána. Pokud by bylo potřeba podrobnějšího popisu, tak je uveden anglicky ve firemní dokumentaci a to v souboru *„fm357-2.pdf“* v kapitole 6.

```
A    "FM1".CH[1].CHB.CH_RDY      // Je kanál 1 připravený ?
JCN   END
R    "FM1".CH[1].CTR.MODE_A      // Vypni automatický mód
S    "FM1".CH[1].CTR.MODE_JOG    // Vyber Jog mód
S    "FM1".CH[1].CTR.REF_POINT   // Nastav referenční přibližovací bod
S    "AX1".AX[1].CTR.PULSE_EN    // Povol PULSE

S    "USERDB".CTR_EN_AX1         // Povol ovládání pro osu 1
A    "USERDB".CTR_EN_AX1         // Je již povoleno ovládání?
=    "AX1".AX[1].CTR.CTR_EN      // Jestliže ano, tak nastav AX-DB osy 1
A    "FM1".CH[1].CHB.MODE_JOG    // Je Jog aktivní ?
```

```
    A    "FM1".CH[1].CHB.REF_POINT   // Je referenční bod přiblížení aktivní?
    JCN   END
    A    "USERDB".DIR_M_AX1          // Je pohyb v negativním směru vybráno ?
    =    "AX1".AX[1].CTR.DIR_M       // Tak zapsat do DB pro osu 1
    A    "USERDB".DIR_P_AX1          // Je pohyb v pozitivním  směru vybráno ?
    =    "AX1".AX[1].CTR.DIR_P       // Tak zapsat do DB pro osu 1
    A    "AX1".AX[1].CHB.GO_M        // Pohyb v negativním směru
    =    "USERDB".GO_M_AX1           // Ulož to do USERDB
    A    "AX1".AX[1].CHB.GO_P        // Pohyb v pozitivním směru
    =    "USERDB".GO_P_AX1           // Ulož to do USERDB
    R    "AX1".AX[1].CTR.FD_STOP     // Smyčka STOP musí být vždy 0
END:
```

8. Závěr

V závěru této diplomové práce bych chtěl zmínit, že tento řídící řetězec je realizován jako laboratorní úloha a v praxi by neměl takové využití, jako model řízený ve třech osách, který by sám hledal největší intenzitu slunečních paprsků. Tato úloha však bude sloužit pro laboratoř ASŘ, kde studenti budou získávat praktické znalosti s ovládáním PLC a použití solárních panelů.

Nastudování problematiky solárních panelů nebyla obtížné, existuje několik dostupných prací na toto téma, včetně diplomových. Mnohem větší problém nastal při samotné realizaci otáčení krokového motoru. Firma Siemens neposlala s moduly žádnou dokumentaci a pouze odkázala na v literatuře zmíněné WWW stránky. Veškerá dostupná dokumentace je pouze ve světových jazycích. Tedy veškerá mnou studovaná literatura byla v anglickém jazyce. Nebylo vůbec jednoduché se pročíst veškerou literaturou a toto samotné studování mi zabralo nejvíce času (celý semestr). Dlouhou dobu trvalo samotné zapojení, protože čtení literatury šlo velice pomalu. Když bylo již vše zapojeno, nastala druhá část a to samotné programování. Instalace softwaru byla jednoduchá, ale jeho obsluha bez manuálu byla dosti složitá. Ale po několika cílených konzultacích jsem pochopil systém samotného programování a pustil se do realizace programu. Nebylo to jednoduché, neboť skládání jednotlivých bloků za sebou muselo být předem velice dobře promyšlené, aby vše dělalo to, co má. Po zvládnutí všech bloků již stačilo pouze roztočit programově krokový motor. Ale zde nastala nejtěžší část celého programování. Na toto neexistuje žádný blok, vše je programováno pomocí nastavování proměnných na „true“ nebo „false“ a nastudovat, jaká proměnná udělá jakou věc nebylo opravdu jednoduché. V literatuře je tomu věnováno cca. 30 stánek příkazů. Ale po prostudování přiložených příkladů jsem použil část připraveného kódu, který firma Siemens doporučuje použít, a pak pouze vpisoval příkazy, které jsem potřeboval. K samotnému roztočení motoru již došlo zanedlouho. Celý program pak byl několik týdnů laděn a zkoušen, zda-li vše správně funguje. Po jeho odzkoušení byl řádně uložen a připraven pro pozdější použití v laboratoři.

Zde bych se rád zmínil, že problematika solárních panelů je v poslední době docela diskutovaná. Vzhledem ke zvýšenému používání čistých energií, je nastudování solárních panelů pro výuku studentů vhodně zvoleno. Také použití PLC pro řízení otáčení rokového motoru není špatná volba. V poslední době se v každém podniku používá automatické řízení a jedním z nejpoužívanějších jednotek pro toto řízení jsou právě programovatelné

automaty. Zkušenost s těmito automaty je velice dobrý základ pro pozdější uplatnění v provozech, které automatizaci teprve zakládají, či již dlouhodobě využívají.

V laboratoři ASŘ by studenti mohli na tomto modelu zkoušet nastavování solárního panelu směrem ke slunci. Tento panel by potom mohli měřit a získávat reálné hodnoty napětí, které dodává panel při určitém osvětlení.

Použitá literatura

[1] Olehla, P. – Netradiční zdroje energie – fotovoltaické články

[2] Lenk, V.,Moučka, M. - Laboratorní měření na solárním panelu

[3] Kobzová, E. - Počasí

[4] Martinásková, M.,Šmejkal, L. – Řízení programovatelnými automaty

[5] Siemens DE - www.ad.siemens.com

Přílohy

V příloze je celý program zkopírovaný ze STEP 7 v nezměněné formě. Tedy i s českými a anglickými popisky. Vkládám pouze program z bloku DB1, protože program z bloku FC100 je již celý v kapitole 7.2.13.

```
      CALL  "BFCT"                 // Basic functions and operating modes of FM 357-2
       FMDB_NO:=30                                  // data block number for FM357-2
signals
      A     "FM1".STARTUP                          // Startup of first FM 357-2
      JC    END                                    // no user program when startup
routine
      A     "FM1".CONNECT                          // Ready to communicate ?
      JCN   END                                    // End if not ready to communicate
      L     B#16#26                                // 100 % Override
      T     "FM1".CH[1].CTR.FD_OVERR               // Feed override channel 1
      T     "AX1".AX[1].CTR.OVERR                  // Override axis 1
      S     "FM1".CH[1].CTR.FD_OVERR_FDR  // Path override feedrate operative
      S     "AX1".AX[1].CTR.ACT_OVERR              // Activate override
      CALL  FC100
      CALL  "READ_CLK"
       RET_VAL:=DB1.DBW8
       CDT    :=DB1.cas_datum
      NOP   0
      CALL  "DT_DATE"
       IN     :=DB1.cas_datum
       RET_VAL:=DB1.DBW14
      NOP   0
      CALL  "DT_TOD"
       IN     :=DB1.cas_datum
       RET_VAL:=DB1.DBD10
      NOP   0
      AN    DB2.DBX   18.1
      JNB   o9
      L     DB3.DBD    4
      L     L#15
      +D
      T     DB3.DBD   16
      AN    OV
      SAVE
      CLR
o9:   A     BR
      S     DB2.DBX   18.1
      AN    I    125.7
      A(
      L     DB2.DBW    0
      L     0
      ==I
```

```
      )
      S     "USERDB".DIR_M_AX1
      A     I    125.7
      JNB   oli
      L     1
      T     DB2.DBW    0
      SET
      SAVE
      CLR
oli:  A     BR
      R     "USERDB".DIR_M_AX1
      A(
      A(
      AN    I    125.6
      JNB   ol1
      L     0
      T     DB2.DBW    0
      SET
      SAVE
      CLR
ol1:  A     BR

      )
      JNB   ol2
      L     DW#16#18E4120
      T     DB3.DBD    4
      SET
      SAVE
      CLR
ol2:  A     BR
      )
      JNB   ol3
      L     DW#16#18E4130
      T     DB3.DBD   16
      SET
      SAVE
      CLR
ol3:  A     BR
      S     Q    125.6
      R     DB2.DBX    2.1
      A(
      L     DB2.DBW    0
      L     1
      ==I
      )
      A(
      L     DB1.DBD   10
      L     L#25200000
      >=D
      )
```

```
      A(
      L    DB1.DBD   10
      L    L#57600000
      <D
      )
      =    Q    125.0
      =    Q    125.2
      A(
      L    DB2.DBW    0
      L    1
      ==I
      )
      A(
      L    DB1.DBD   10
      L    DB3.DBD    4
      >D
      )
      A(
      L    DB1.DBD   10
      L    DB3.DBD   16
      <D
      )
      S    "USERDB".DIR_P_AX1
      A    "AX1".AX[1].CTR.DIR_P
      A    DB2.DBX    2.1
      AN   DB2.DBX   18.0
      L    S5T#20MS
      SD   T     1
      A    T     1
      R    "USERDB".DIR_P_AX1
      R    T     1
      S    DB2.DBX    2.0
      A(
      A    DB2.DBX    2.0
      JNB  oli1
      L    DB3.DBD    4
      L    DB3.DBD   12
      +D
      T    DB3.DBD    4
      AN   OV
      SAVE
      CLR
oli1: A    BR
      )
      JNB  oli2
      L    DB3.DBD   16
      L    DB3.DBD   12
      +D
      T    DB3.DBD   16
      AN   OV
```

```
      SAVE
      CLR
oli2: A     BR
      R     DB2.DBX    2.0
      A(
      A(
      A(
      L     DB1.DBD   10
      L     L#57600500
      >=D
      )
      A(
      L     DB1.DBD   10
      L     L#57601000
      <=D
      )
      JNB   oli3
      L     0
      T     DB2.DBW    0
      SET
      SAVE
      CLR
oli3: A     BR
      )
      JNB   oli4
      L     DW#16#18E4120
      T     DB3.DBD    4
      SET
      SAVE
      CLR
oli4: A     BR
      )
      JNB   oli5
      L     DW#16#18E4135
      T     DB3.DBD   16
      SET
      SAVE
      CLR
oli5: A     BR
      R     DB2.DBX    2.1
      R     DB2.DBX   18.1
      A(
      A(
      A(
      A(
      AN    DB2.DBX    2.1
      A     I    125.6
      A(
      L     DB2.DBW    0
      L     1
```

```
    ==I
    )
    A(
    L    DB1.DBD   10
    L    L#25200000
    >=D
    )
    A(
    L    DB1.DBD   10
    L    L#57600500
    <D
    )
    JNB   oli6
    L    DB1.DBD   10
    L    L#25200000
    -D
    T    DB2.DBD    4
    AN    OV
    SAVE
    CLR
oli6: A     BR
    )
    JNB   oli7
    L    DB2.DBD    4
    L    L#900000
    /D
    T    DB2.DBD    4
    AN    OV
    SAVE
    CLR
oli7: A     BR
    )
    JNB   oli8
    L    DB2.DBD    4
    L    L#900000
    *D
    T    DB2.DBD    4
    AN    OV
    SAVE
    CLR
oli8: A     BR
    )
    JNB   oli9
    L    DB2.DBD    4
    L    DB3.DBD    4
    +D
    T    DB3.DBD    4
    AN    OV
    SAVE
    CLR
```

```
oli9: A     BR
      )
      JNB   oliv
      L     DB2.DBD    4
      L     DB3.DBD   16
      +D
      T     DB3.DBD   16
      AN    OV
      SAVE
      CLR
oliv: A     BR
      S     DB2.DBX   12.0
      S     DB2.DBX    2.1
      A(
      A(
      A(
      A(
      A     DB2.DBX   12.0
      JNB   o1
      L     DB1.DBD   10
      T     DB3.DBD   20
      SET
      SAVE
      CLR
o1:   A     BR
      )
      JNB   o2
      L     DB3.DBD   20
      T     DB3.DBD   24
      SET
      SAVE
      CLR
o2:   A     BR
      )
      JNB   o3
      L     DB2.DBD    4
      L     L#900000
      /D
      T     DB2.DBD    8
      AN    OV
      SAVE
      CLR
o3:   A     BR
      )
      JNB   o4
      L     DB2.DBD    8
      L     L#20
      *D
      T     DB2.DBD    8
      AN    OV
```

```
    SAVE
    CLR
o4:  A    BR
    )
    JNB   o5
    L    DB3.DBD   24
    L    DB2.DBD    8
    +D
    T    DB3.DBD   24
    R    DB2.DBX   12.0
    S    DB2.DBX   18.0
o5:  NOP   0
    A    DB2.DBX   18.0
    A(
    L    DB1.DBD   10
    L    DB3.DBD   20
    >=D
    )
    A(
    L    DB1.DBD   10
    L    DB3.DBD   24
    <=D
    )
    S    "USERDB".DIR_P_AX1
    A    DB2.DBX   18.0
    A(
    L    DB1.DBD   10
    L    DB3.DBD   24
    >D
    )
    R    "USERDB".DIR_P_AX1
    R    DB2.DBX   18.0
END:  BE
```